



#2
6-29-99
JC398 U.S. PTO
09/112786
07/10/98


Patent Office
Canberra

I, MARIO PERUSSICH, ASSISTANT DIRECTOR PATENT SERVICES, hereby certify that the annexed is a true copy of the Provisional specification in connection with Application No. PO 8504 for a patent by SILVERBROOK RESEARCH PTY LTD filed on 11 August 1997.

I further certify that the annexed specification is not, as yet, open to public inspection.

WITNESS my hand this Nineteenth
day of June 1998

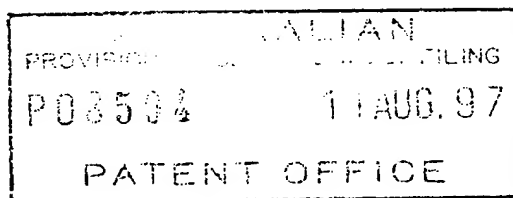
5


MARIO PERUSSICH
ASSISTANT DIRECTOR PATENT SERVICES

P/00/009
Regulation 3.2

AUSTRALIA
Patents Act 1990

PROVISIONAL SPECIFICATION



Application Title: Image Processing Method and Apparatus (ART42)

The invention is described in the following statement:

GH REF: 23975DY/PJT

A Method of Manufacture of an Image Creation Apparatus (ART 42)

Field of the Invention

The present invention relates to digital image processing and in particular discloses Camera System
5 Containing a VLIW Vector Processor.

Further the present invention relates to an image processing method and apparatus and, in particular, discloses a Digital Instant Camera with Image Processing Capability.

10 The present invention further relates to the field of digital camera technology and, particularly, discloses a digital camera having an integral color printer.

Background of the Invention

Traditional camera technology has for many years relied
15 upon the provision of an optical processing system which relies on a negative of an image which is projected onto a photosensitive film which is subsequently chemically processed so as to "fix" the film and to allow for positive prints to be produced which reproduce the original image.
20 Such an image processing technology, although it has become a standard, can be unduly complex, as expensive and difficult technologies are involved in full color processing of images. Recently, digital cameras have become available. These cameras normally rely upon the utilization of a
25 charged coupled device (CCD) to sense a particular image. The camera normally includes storage media for the storage of the sensed scenes in addition to a connector for the transfer of images to a computer device for subsequent manipulation and printing out.

30 Such devices are generally inconvenient in that all images must be stored by the camera and printed out at some later stage. Hence, the camera must have sufficient storage capabilities for the storing of multiple images and, additionally, the user of the camera must have access to a
35 subsequent computer system for the downloading of the images and printing out by a computer printer or the like.

Further, digital camera devices have only limited on board processing capabilities which can only perform limited manipulation of sensed image. The main function of the on board processing capability is to store the sensed image.

5 As it may be desirable to carry out extensive modification of an image, the capabilities of such digital camera devices are considered inadequate.

Summary of the Invention

10 The present invention relates to the provision of a digital camera system having significant on-board computational capabilities for the manipulation of images.

In accordance with a first aspect of the present invention, there is provided a digital camera system comprising a sensing means for sensing an image;

15 modification means for modifying the sensed image in accordance with modification instructions input into the camera; and an output means for outputting the modified image; wherein the modification means includes a series of processing elements arranged around a central crossbar
20 switch. Preferably, the processing elements include an Arithmetic Logic Unit (ALU) acting under the control of a microcode store wherein the microcode store comprises a writeable control store. The processing elements can include an internal input and output FIFO for storing
25 pixel data utilized by the processing elements and the modification means is interconnected to a read and write FIFO for reading and writing pixel data of images to the modification means.

Each of the processing elements can be arranged in a
30 ring and each element is also separately connected to its nearest neighbours. The ALU accepts a series of inputs interconnected via an internal crossbar switch to a series of core processing units within the ALU and includes a number of internal registers for the storage of temporary
35 data. The core processing units can include at least one one of a multiplier, an adder and a barrel shifter.

The processing elements are further connected to a common data bus for the transfer of pixel data to the processing elements and the data bus is interconnected to a data cache which acts as an intermediate cache between the processing elements and a memory store for storing the images.

Brief Description of the Drawings

Notwithstanding any other forms which may fall within the scope of the present invention, preferred forms of the invention will now be described, by way of example only, with reference to the accompanying drawings in which:

Fig. 1 illustrates an Artcam device constructed in accordance with the preferred embodiment;

Fig. 2 is a schematic block diagram of the main Artcam electronic components;

Fig. 3 is a schematic block diagram of the Artcam Central Processor;

Fig. 4 illustrates the CCD image organization;

Fig. 5 illustrates the storage format for a logical image;

Fig. 6 illustrates the internal image memory storage format;

Fig. 7 illustrates the image pyramid storage format;

Fig. 8 illustrates the process steps in creating an output image;

Fig. 9 illustrates the operation of an image iterator;

Fig. 10 illustrates an example read iterator;

Fig. 11 illustrates a standard pixel process;

Fig. 12 illustrates a pixel reading process;

Fig. 13 illustrates a first example box read iterator output;

Fig. 14 illustrates a second example box read iterator output;

Fig. 15 illustrates a Box Read Iterator Process;

Fig. 16 illustrates the storage format utilised by a vertical strip Iterator;

Fig. 17 illustrates a process that requires only a vertical strip Write Iterator;

Fig. 18 illustrates in schematic form the VLIW vector processor;

5 Fig. 19 illustrates in schematic form the ALU unit in more detail;

Fig. 20 illustrates in schematic form of the ALU proper;

Fig. 21 illustrates the structure of ALU input block;

10 Fig. 22 illustrates the structure of the Latch2 block;

Fig. 23 illustrates the read process block;

Fig. 24 illustrates the Barrel Shifter process block;

Fig. 25 illustrates the structure of the Adder/Logic Block;

15 Fig. 26 illustrates the structure of the Latches₁ process block;

Fig. 27 illustrates the internal structure of the multiply/interpolate process block;

20 Fig. 28 illustrates the process of generating a sequential read;

Fig. 29 illustrates the internal portion of the sequential coordinate generator;

Fig. 30 illustrates the vertical strip generation process;

25 Fig. 31 illustrates an implementation of the vertical strip generation process;

Fig. 32 illustrates the structure of the Display Controller;

30 Fig. 33 illustrates a time line of the process of sampling an Artcard;

Fig. 34 illustrates the super sampling process;

Fig. 35 illustrates the process of reading a rotated Artcard;

35 Fig. 36 illustrates a flow chart of the steps necessary to decode an Artcard;

Fig. 37 illustrates an enlargement of the left hand

corner of a single Artcard;

Fig. 38 illustrates a single target for detection;

Fig. 39 illustrates the method utilised to detect targets;

5 Fig. 40 illustrates the method of calculating the distance between two targets;

Fig. 41 illustrates the process of centroid drift;

Fig. 42 shows one form of centroid lookup table;

Fig. 43 illustrates the centroid updating process;

10 Fig. 44 illustrates a delta processing lookup table utilised in the preferred embodiment;

Fig. 45 illustrates the process of unscrambling Artcard data;

15 Fig. 46 illustrates one form of implementation of the convolver;

Fig. 47 illustrates a convolution process;

Fig. 48 illustrates the compositing process;

Fig. 49 illustrates the regular compositing process in more detail;

20 Fig. 50 illustrates the process of warping using a warp map;

Fig. 51 illustrates the warping bi-linear interpolation process;

Fig. 52 illustrates the process of span calculation;

25 Fig. 53 illustrates the basic span calculation process;

Fig. 54 illustrates one form of detail implementation of the span calculation process;

Fig. 55 illustrates the process of reading image pyramid levels;

30 Fig. 56 illustrates using the pyramid table for blinear interpolation;

Fig. 57 illustrates the histogram collection process;

Fig. 58 illustrates the color transform process;

Fig. 59 illustrates the color conversion process;

35 Fig. 60 illustrates the color space conversion process in more detail;

Fig. 61 illustrates the process of calculating an input coordinate;

Fig. 62 illustrates the process of compositing with feedback;

5 Fig. 63 illustrates the generalized scaling process;

Fig. 64 illustrates the scale in X scaling process;

Fig. 65 illustrates the scale in Y scaling process;

Fig. 66 illustrates the tessellation process;

Fig. 67 illustrates the sub-pixel translation process;

10 Fig. 68 illustrates the compositing process;

Fig. 69 illustrates the process of compositing with feedback;

Fig. 70 illustrates the process of tiling with color from the input image;

15 Fig. 71 illustrates the process of tiling with feedback;

Fig. 72 illustrates the process of tiling with texture replacement;

20 Fig. 73 illustrates the process of tiling with color from the input image;

Fig. 74 illustrates the process of applying a texture without feedback;

Fig. 75 illustrates the process of applying a texture with feedback;

25 Fig. 76 illustrates the process of rotation of CCD pixels;

Fig. 77 illustrates the process of interpolation of Green subpixels;

30 Fig. 78 illustrates the process of interpolation of Blue subpixels;

Fig. 79 illustrates the process of interpolation of Red subpixels;

Fig. 80 illustrates the process of CCD pixel interpolation with 0 degree rotation for odd pixel lines;

35 Fig. 81 illustrates the process of CCD pixel interpolation with 0 degree rotation for even pixel lines;

Fig. 82 illustrates the process of color conversion to Lab color space;

Fig. 83 illustrates the process of calculation of $1/\sqrt{X}$;

Fig. 84 illustrates the implementation of the
5 calculation of $1/\sqrt{X}$ in more detail;

Fig. 85 illustrates the process of Normal calculation with a bump map;

Fig. 86 illustrates the process of illumination calculation with a bump map;

10 Fig. 87 illustrates the process of illumination calculation with a bump map in more detail;

Fig. 88 illustrates the process of calculation of L using a directional light;

Fig. 89 illustrates the process of calculation of L
15 using a Omni lights and spotlights;

Fig. 90 illustrates one form of implementation of calculation of L using a Omni lights and spotlights;

Fig. 91 illustrates the process of calculating the N.L dot product;

20 Fig. 92 illustrates the process of calculating the N.L dot product in more detail;

Fig. 93 illustrates the process of calculating the R.V dot product;

Fig. 94 illustrates the process of calculating the R.V
25 dot product in more detail;

Fig. 95 illustrates the attenuation calculation inputs and outputs; *

Fig. 96 illustrates an actual implementation of attenuation calculation;

30 Fig. 97 illustrates an graph of the cone factor;

Fig. 98 illustrates the process of penumbra calculation;

Fig. 99 illustrates the angles utilised in penumbra calculation;

35 Fig. 100 illustrates the inputs and outputs to penumbra calculation;

Fig. 101 illustrates an actual implementation of penumbra calculation;

Fig. 102 illustrates the inputs and outputs to ambient calculation;

5 Fig. 103 illustrates an actual implementation of ambient calculation;

Fig. 104 illustrates an actual implementation of diffuse calculation;

10 Fig. 105 illustrates the inputs and outputs to a diffuse calculation;

Fig. 106 illustrates an actual implementation of a diffuse calculation;

Fig. 107 illustrates the inputs and outputs to a specular calculation;

15 Fig. 108 illustrates an actual implementation of a specular calculation;

Fig. 109 illustrates the inputs and outputs to a specular calculation;

20 Fig. 110 illustrates an actual implementation of a specular calculation;

Fig. 111 illustrates an actual implementation of a ambient only calculation;

Fig. 112 illustrates the process overview of light calculation;

25 Fig. 113 illustrates an example illumination calculation for a single infinite light source;

Fig. 114 illustrates an example illumination calculation for a Omni light source without a bump map;

30 Fig. 115 illustrates an example illumination calculation for a Omni light source with a bump map;

Fig. 116 illustrates an example illumination calculation for a Spotlight light source without a bump map;

Fig. 117 illustrates the process of applying a single Spotlight onto an image with an associated bump-map;

35 Fig. 118 illustrates the logical layout of a single printhead;

Fig. 119 illustrates the structure of the printhead interface;

Fig. 120 illustrates the process of rotation of a Lab image;

5 Fig. 121 illustrates the format of a pixel of the printed image;

Fig. 122 illustrates the dithering process;

Fig. 123 illustrates the process of generating an 8 bit dot output;

10 Fig. 124 illustrates a perspective view of the card reader;

Fig. 125 illustrates an exploded perspective of a card reader;

15 Fig. 126 illustrates a close up view of the Artcard reader;

Fig. 127 illustrates a perspective view of the print roll and print head;

Fig. 128 illustrates a first exploded perspective view of the print roll;

20 Fig. 129 illustrates a second exploded perspective view of the print roll;

Fig. 130 illustrates the print roll authentication chip;

25 Fig. 131 illustrates an enlarged view of the print roll authentication chip;

Fig. 132 illustrates the architecture of the print roll authentication*chip;

Fig. 133 sets out the information stored on the print roll authentication chip;

30 Fig. 134 illustrates the authentication process upon insertion of a print roll;

Fig. 135 illustrates a shielding metal layer placed on top of the authentication chip;

35 Fig. 136 illustrates the data stored within the Artcam authorization chip;

Fig. 137 illustrates the process of print head pulse

characterization;

Fig. 138 is an exploded perspective, in section, of the print head ink supply mechanism;

Fig. 139 is a bottom perspective of the ink head supply
5 unit;

Fig. 140 is a bottom side sectional view of the ink head supply unit;

Fig. 141 is a top perspective of the ink head supply unit;

10 Fig. 142 is a top side sectional view of the ink head supply unit;

Fig. 143 illustrates a perspective view of a small portion of the print head;

Fig. 144 illustrates is an exploded perspective of the
15 print head unit;

Fig. 145 illustrates a top side perspective view of the internal portions of an Artcam camera, showing the parts flattened out;

Fig. 146 illustrates a bottom side perspective view of
20 the internal portions of an Artcam camera, showing the parts flattened out;

Fig. 147 illustrates a first top side perspective view of the internal portions of an Artcam camera, showing the parts as encased in an Artcam;

25 Fig. 148 illustrates a second top side perspective view of the internal portions of an Artcam camera, showing the parts as encased in an Artcam;

Fig. 149 illustrates a second top side perspective view of the internal portions of an Artcam camera, showing the
30 parts as encased in an Artcam;

Fig. 150 illustrates the backing portion of a postcard print roll;

Fig. 151 illustrates the corresponding front image on the postcard print roll after printing out images;

35 Fig. 152 illustrates a form of print roll ready for purchase by a consumer.

Description of Preferred and Other Embodiments

The digital image processing camera system constructed in accordance with the preferred embodiment is as illustrated in Fig. 1. The camera unit 1 includes means for the insertion of an integral print roll (not shown). The camera unit 1 can include an area image sensor 2 which sensors an image 3 for captured by the camera. Optionally, the second area image sensor can be provided to also image the scene 3 and to optionally provide for the production of stereographic output effects.

The camera 1 can include an optional color display 5 for the display of the image being sensed by the sensor 2. When a simple image is being displayed on the display 5, the button 6 can be depressed resulting in the printed image 8 being output by the camera unit 1. A series of cards, herein after known as "Artcards" 9 contain, on one surface encoded information and on the other surface, contain an image distorted by the particular effect produced by the Artcard 9. The Artcard 9 is inserted in an Artcard reader 10 in the side of camera 1 and, upon insertion, results in output image 8 being distorted in the same manner as the distortion appearing on the surface of Artcard 9. Hence, by means of this simple user interface a user wishing to produce a particular effect can insert one of many Artcards 9 into the Artcard reader 10 and utilize button 19 to take a picture of the image 3 resulting in a corresponding distorted output image 8.

The camera unit 1 can also include a number of other control button 13, 14 in addition to a simple LCD output display 15 for the display of informative information including the number of printouts left on the internal print roll on the camera unit. Additionally, different output formats can be controlled by CHP switch 17.

Turning now to Fig. 2, there is illustrated a schematic view of the internal hardware of the camera unit 1. The internal hardware is based around an Artcam central

processor unit (ACP) 31.

Artcam Central Processor 31

The Artcam central processor 31 provides many functions which form the 'heart' of the system. The ACP 31 is preferably implemented as a complex, high speed, CMOS system on-a-chip. Utilising standard cell design with some full custom regions is recommended. Fabrication on a 0.25 μ CMOS process will provide the density and speed required, along with a reasonably small die area.

The functions provided by the ACP 31 include:

1. Control and digitization of the area image sensor
2. A 3D stereoscopic version of the ACP requires two area image sensor interfaces with a second optional image sensor 4 being provided for stereoscopic effects.
2. Area image sensor compensation, reformatting, and image enhancement.
3. Memory interface and management to a memory store 33.
4. Interface, control, and analog to digital conversion of an Artcard reader linear image sensor 34 which is provided for the reading of data from the Artcards 9.
5. Extraction of the raw Artcard data from the digitized and encoded Artcard image.
6. Reed-Solomon error detection and correction of the Artcard encoded data. The encoded surface of the Artcard 9 includes information on how to process an image to produce the effects displayed on the image distorted surface of the Artcard 9. This information is in the form of a script, hereinafter known as a "Vark script". The Vark script is utilised by an interpreter running within the ACP 31 to produce the desired effect.
7. Interpretation of the Vark script on the Artcard 9.
8. Performing image processing operations as specified by the Vark script.

9. Controlling various motors for the paper transport 36, zoom lens 38, autofocus 39 and Artcard driver 37.

10. Controlling a guillotine actuator 40 for the operation of a guillotine 41 for the cutting of photographs 8 from print roll 42.

11. Half-toning of the image data for printing.

12. Providing the print data to a print-head 44 at the appropriate times.

13. Controlling the print head 44.

10 14. Controlling the ink pressure feed to print-head 44.

15. Controlling optional flash unit 56.

16. Reading and acting on various sensors in the camera, including camera orientation sensor 46, autofocus 47 and Artcard insertion sensor 49.

17. Reading and acting on the user interface buttons 6, 13, 14.

18. Controlling the status display 15.

20 19. Providing viewfinder and preview images to the color display 5.

20. Control of the system power consumption, including the ACP power consumption via power management circuit 51 .

21. Providing external communications 52 to general purpose computers (using part USB).

25 22. Reading and storing information in a printing roll authentication chip 53.

23. Reading and storing information in a camera authentication chip 54.

30 24. Communicating with an optional mini-keyboard 57 for text modification.

Quartz crystal 58

A quartz crystal 58 is used as a frequency reference for the system clock. As the system clock is very high, the ACP 31 includes a phase locked loop clock circuit to increase the frequency derived from the crystal 58.

Image Sensing

Area image sensor 2

The area image sensor 2 converts an image through its lens into an electrical signal. It can either be a charge coupled device (CCD) or an active pixel sensor (APS) CMOS image sensor. At present, available CCD's normally have a higher image quality, however, there is currently much development occurring in CMOS imagers. CMOS imagers are eventually expected to be substantially cheaper than CCD's have smaller pixel areas, and be able to incorporate drive circuitry and signal processing. They can also be made in CMOS fabs, which are transitioning to 12" wafers. CCD's are usually built in 6" wafer fabs, and economics may not allow a conversion to 12" fabs. Therefore, the difference in fabrication cost between CCD's and CMOS imagers is likely to increase, progressively favoring CMOS imagers. However, at present, a CCD is probably the best option.

The Artcam unit will produce suitable results with a 1,500 x 1,000 area image sensor. However, smaller sensors, such as 750 x 500, will be adequate for many markets. The Artcam is less sensitive to image sensor resolution than are conventional digital cameras. This is because many of the styles contained on Artcards 9 process the image in such a way as to obscure the lack of resolution. For example, if the image is distorted to simulate the effect of being converted to an impressionistic painting, low source image resolution can be used with minimal effect. Further examples for which low resolution input images will typically not be noticed include image warps which produce high distorted images, multiple miniature copies of the of the image (eg. passport photos), textural processing such as bump mapping for a base relief metal look, and photo-compositing into structured scenes.

This tolerance of low resolution image sensors may be a significant factor in reducing the manufacturing cost of an Artcam unit 1 camera. An Artcam with a low cost 750 x 500

image sensor will often produce superior results to a conventional digital camera with a much more expensive 1,500 x 1,000 image sensor.

Optional stereoscopic 3D image sensor 4

5 The 3D versions of the Artcam unit 1 have an additional image sensor 4, for stereoscopic operation. This image sensor is identical to the main image sensor. The circuitry to drive the optional image sensor may be included as a standard part of the ACP chip 31 to reduce incremental
10 design cost. Alternatively, a separate 3D Artcam ACP can be designed. This option will reduce the manufacturing cost of a mainstream single sensor Artcam.

Print roll authentication chip 53

15 A small chip 53 is included in each print roll 42. This chip replaced the functions of the bar code, optical sensor and wheel, and ISO/ASA sensor on other forms of camera film units such as Advanced Photo Systems film cartridges.

 The authentication chip also provides other features:

- 20 1. The storage of data rather than that which is mechanically and optically sensed from APS rolls
2. A remaining media length indication, accurate to high resolution.
3. Authentication Information to prevent inferior
25 clone print roll copies.

 The authentication chip 53 contains 1024 bits of Flash memory, of which 128 bits is an authentication key, and 512 bits is the authentication information. Also included is an encryption circuit to ensure that the authentication key
30 cannot be accessed directly.

Print-head 44

 The Artcam unit 1 can utilize any color print technology which is small enough, low enough power, fast enough, high enough quality, and low enough cost, and is
35 compatible with the print roll. Relevant printheads will be specifically discussed hereinafter.

The specifications of the ink jet head are:

Image type	Bi-level, dithered
Color	CMY Process Color
Resolution	1600 dpi
Print head length	'Page-width' (100mm)
Print speed	2 seconds per photo

Optional ink pressure Controller (not shown)

5 The function of the ink pressure controller depends
upon the type of ink jet print head 44 incorporated in the
Artcam. For some types of ink jet, the use of an ink
pressure controller can be eliminated, as the ink pressure
is simply atmospheric pressure. Other types of print head
10 require a regulated positive ink pressure. In this case,
the in pressure controller consists of a pump and pressure
transducer.

Other print heads may require an ultrasonic transducer
to cause regular oscillations in the ink pressure, typically
at frequencies around 100KHz. In the case, the APC 31
15 controls the frequency phase and amplitude of these
oscillations.

Paper transport motor 36

The paper transport motor 36 moves the paper from
within the print roll 42 past the print head at a relatively
20 constant rate. The motor 36 is a miniature motor geared
down to an appropriate speed to drive rollers which move the
paper. A high quality motor and mechanical gears are
required to achieve high image quality, as mechanical rumble
or other vibrations will affect the printed dot row spacing.

25 Paper transport motor driver 60

The motor driver 60 is a small circuit which amplifies
the digital motor control signals from the APC 31 to levels
suitable for driving the motor 36.

Paper pull sensor

30 A paper pull sensor 50 detects a user's attempt to pull
a photo from the camera unit during the printing process.

The APC 31 reads this sensor 50, and activates the guillotine 41 if the condition occurs. The paper pull sensor 50 is incorporated to make the camera more 'foolproof' in operation. Were the user to pull the paper out forcefully during printing, the print mechanism 44 or print roll 42 may (in extreme cases) be damaged. Since it is acceptable to pull out the 'pod' from a Polaroid type camera before it is fully ejected, the public has been 'trained' to do this. Therefore, they are unlikely to heed printed instructions not to pull the paper.

The Artcam preferably restarts the photo print process after the guillotine 41 has cut the paper after pull sensing.

The pull sensor can be implemented as a strain gauge sensor, or as an optical sensor detecting a small plastic flag which is deflected by the torque that occurs on the paper drive rollers when the paper is pulled. The latter implementation is recommendation for low cost.

Paper guillotine actuator

20 Paper guillotine actuator 40

The paper guillotine actuator 40 is a small actuator which causes the guillotine 41 to cut the paper either at the end of a photograph, or when the paper pull sensor 50 is activated.

25 The guillotine actuator 40 is a small circuit which amplifies a guillotine control signal from the APC tot the level required by the actuator 41.

Artcard 9

30 The Artcard 9 is a program storage medium for the Artcam unit. As noted previously, the programs are in the form of Vark scripts. Vark is a powerful image processing language especially developed for the Artcam unit. Each Artcard 9 contains one Vark script, and thereby defines one image processing style.

35 Preferably, the VARK language is highly image processing specific. By being highly image processing

specific, the amount of storage required to store the details on the card are substantially reduced. Further, the ease with which new programs can be created, including enhanced effects, is also substantially increased.

5 Preferably, the language includes facilities for handling many image processing functions including image warping via a warp map, convolution, color lookup tables, posturizing an image, adding noise to an image, image enhancement filters, painting algorithms, brush jittering and manipulation edge
10 detection filters, tiling, illumination via light sources, bump maps, text, face detection and object detection attributes, fonts, including three dimensional fonts, and arbitrary complexity pre-rendered icons.

Hence, by utilizing the language constructs as defined
15 by the created language, new affects on arbitrary images can be created and constructed for inexpensive storage on Artcard and subsequent distribution to camera owners. Further, on one surface of the card can be provided an example illustrating the effect that a particular VARK
20 script, stored on the other surface of the card, will have on an arbitrary captured image.

By utilizing such a system, camera technology can be distributed without a great fear of obsolescence in that, provided a VARK interpreter is incorporated in the camera
25 device, a device independent scenario is provided whereby the underlying technology can be completely varied over time. Further, the VARK scripts can be updated as new filters are created and distributed in an inexpensive manner, such as via simple cards for card reading.

30 The Artcard 9 is a piece of thin white plastic with the same format as a credit card (86mm long by 54mm wide). The Artcard is printed on both sides using a high resolution ink jet printer. The inkjet printer technology is assumed to be the same as that used in the Artcam, with 1600 dpi (63dpm) resolution.
35 A major feature of the Artcard 9 is low manufacturing cost. Artcards can be manufactured at high

speeds as a wide web of plastic film. The plastic web is coated on both sides with a hydrophilic dye fixing layer. The web is printed simultaneously on both sides using a 'pagewidth' color ink jet printer. The web is then cut and punched into individual cards. On one face of the card is printed a human readable representation of the effect the Artcard 9 will have on the sensed image. This can be simply a standard image which has been processed using the Vark script stored on the back face of the card.

On the back face of the card is printed an array of dots which can be decoded into the Vark script that defines the image processing sequence. The print area is 80mm x 50mm, giving a total of 15,876,000 dots. This array of dots could represent at least 1.89 Mbytes of data. To achieve high reliability, extensive error detection and correction is incorporated in the array of dots. This allows a substantial portion of the card to be defaced, worn, creased, or dirty with no effect on data integrity. The data coding used is Reed-Solomon coding, with half of the data devoted to error correction. This allows the storage of 967 Kbytes of error corrected data on each Artcard 9.

Linear image sensor 34

The Artcard linear sensor 34 converts the aforementioned Artcard data image to electrical signals. As with the area image sensor 2, 4, the linear image sensor can be fabricated using either CCD or APS CMOS technology. The active length of the image sensor 34 is 50mm, equal to the width of the data array on the Artcard 9. To satisfy Nyquist's sampling theorem, the resolution of the linear image sensor 34 must be at least twice the highest spatial frequency of the Artcard optical image reaching the image sensor. In practice, data detection is easier if the image sensor resolution is substantially above this. A resolution of 4800 dpi (189 dpmm) is chosen, giving a total of 9,450 pixels. This resolution requires a pixel sensor pitch of 5.3 μ m. This can readily be achieved by using four staggered

rows of 20 μ m pixel sensors.

The linear image sensor is mounted in a special package which includes a LED 65 to illuminate the Artcard 9 via a light-pipe (not shown).

5 The Artcard reader light-pipe can be a molded light-pipe which has several function:

1. It diffuses the light from the LED over the width of the card using total internal reflection facets.

2. It focuses the light onto a 16 μ m wide strip of the
10 Artcard 9 using an integrated cylindrical lens.

3. It focuses light reflected from the Artcard onto the linear image sensor pixels using a molded array of microlenses.

15 The operation of the Artcard reader is explained further hereinafter.

Artcard reader motor 37

20 The Artcard reader motor propels the Artcard past the linear image sensor 34 at a relatively constant rate. As it may not be cost effective to include extreme precision mechanical components in the Artcard reader, the motor 37 is
a standard miniature motor geared down to an appropriate speed to drive a pair of rollers which move the Artcard 9. The speed variations, rumble, and other vibrations will affect the raw image data as circuitry within the APC 31
25 includes extensive compensation for these effects to reliably read the Artcard data.

The motor* 37 is driven in reverse when the Artcard is to be ejected.

Artcard motor driver 61

30 The Artcard motor driver 61 is a small circuit which amplifies the digital motor control signals from the APC 31 to levels suitable for driving the motor 37.

Card Insertion sensor 49

35 The card insertion sensor 49 is an optical sensor which detects the presence of a card as it is being inserted in the card reader 34. Upon a signal from this sensor 49, the

APC 31 initiates the card reading process, including the activation of the Artcard reader motor 37.

Card eject button 16

5 A card eject button 16 (Fig. 1) is used by the user to eject the current Artcard, so that another Artcard can be inserted. The APC 31 detects the pressing of the button, and reverses the Artcard reader motor 37 to eject the card.

Card status indicator 66

10 A card status indicator 66 is provided to signal the user as to the status of the Artcard reading process. This can be a standard bi-color (red/green) LED. When the card is successfully read, and data integrity has been verified, the LED lights up green continually. If the card is faulty, then the LED lights up red.

15 If the camera is powered from a 1.5 V instead of 3V battery, then the power supply voltage is less than the forward voltage drop of the green LED, and the LED will not light. In this case, red LEDs can be used, or the LED can be powered from a voltage pump which also powers other
20 circuits in the Artcam which require higher voltage.

64 Mbit DRAM 33

To perform the wide variety of image processing effects, the camera utilizes 8 Mbytes of memory 33. This can be provided by a single 64 Mbit memory chip. Of course,
25 with changing memory technology increased Dram storage sizes may be substituted.

High speed access to the memory chip is required. This can be achieved by using a Rambus DRAM (burst access rate of 500 Mbytes per second) or chips using the new open standards
30 such as double data rate (DDR) SDRAM or Synclink DRAM.

Camera authentication chip

The camera authentication chip 54 is identical to the print roll authentication chip 53, except that it has different information stored in it. The camera
35 authentication chip 54 has three main purposes:

1. To provide a secure means of comparing

authentication codes with the print roll authentication chip;

2. To provide storage for manufacturing information, such as the serial number of the camera;

5 3. To provide a small amount of non-volatile memory for storage of user information.

Displays

The Artcam includes an optional color display 5 and small status display 15. Lowest cost consumer cameras may
10 include a color image display, such as a small TFT LCD 5 similar to those found on some digital cameras and camcorders. The color display 5 is a major cost element of these versions of Artcam, and the display 5 plus back light are a major power consumption drain.

15 Status display 15

The status display 15 is a small passive segment based LCD, similar to those currently provided on silver halide and digital cameras. Its main function is to show the
20 number of prints remaining in the print roll 42 and icons for various standard camera features, such as flash and battery status.

Color display 5

The color display 5 is a full motion image display which operates as a viewfinder, as a verification of the
25 image to be printed, and as a user interface display. The cost of the display 5 is approximately proportional to its area, so large displays (say 4" diagonal) unit will be restricted to expensive versions of the Artcam unit. Smaller displays, such as color camcorder viewfinder TFT's
30 at around 1", may be effective for mid-range Artcams. Zoom lens (not shown)

The Artcam can include a zoom lens. This can be a standard electronically controlled zoom lens, identical to one which would be used on a standard electronic camera, and
35 similar to pocket camera zoom lenses. A referred version of the Artcam unit may include standard interchangeable 35mm

SLR lenses.

Autofocus motor 39

The autofocus motor 39 changes the focus of the zoom lens. The motor is a miniature motor geared down to an appropriate speed to drive the autofocus mechanism.

Autofocus motor driver 63

The autofocus motor driver 63 is a small circuit which amplifies the digital motor control signals from the APC 31 to levels suitable for driving the motor 39.

Zoom motor 38

The zoom motor 38 moves the zoom front lenses in and out. The motor is a miniature motor geared down to an appropriate speed to drive the zoom mechanism.

Zoom motor driver 62

The zoom motor driver 62 is a small circuit which amplifies the digital motor control signals from the APC 31 to levels suitable for driving the motor.

Communications

The ACP 31 contains a universal serial bus (USB) interface 52 for communication with personal computers. Not all Artcam models are intended to include the USB connector. However, the silicon area required for a USB circuit 52 is small, so the interface can be included in the standard ACP.

Optional Keyboard 57

The Artcam unit may include an optional miniature keyboard 57 for customizing text specified by the Artcard. Any text appearing in an Artcard image may be editable, even if it is in a complex metallic 3D font. The miniature keyboard includes a single line alphanumeric LCD to display the original text and edited text. The keyboard may be a standard accessory.

The ACP 31 contains a serial communications circuit for transferring data to and from the miniature keyboard.

Power Supply

The Artcam unit uses a single battery 48. Depending upon the Artcam options, this is either a 3V Lithium cell,

or a 1.5 VAA or AAA alkaline cell.

Power Management Unit 51

Power consumption is an important design constraint in the Artcam. It is desirable that either standard camera
5 batteries (such as 3V lithium batters) or standard AA or AAA alkaline cells can be used. While the electronic complexity of the Artcam unit is dramatically higher than 35mm
photographic cameras, the power consumption need not be commensurately higher. Power in the Artcam can be carefully
10 managed with all unit being turned off when not in use.

The most significant current drains are the ACP 31, the area image sensors 2,4, the printer 44 various motors, the flash unit 56, and the optional color display 5 dealing with each part separately:

15 1. ACP: If fabricated using 0.25 μ m CMOS, and running on 1.5V, the ACP power consumption can be quite low. Clocks to various parts of the ACP chip can be quite low. Clocks to various parts of the ACP chip can be turned off when not in use, virtually eliminating standby current consumption.
20 The ACP will only fully used for approximately 4 seconds for each photograph printed.

2. Area image sensor: power is only supplied to the area image sensor when the user has their finger on the button.

25 3. The printer power is only supplied to the printer when actually printing. This is for around 2 seconds for each photograph. Even so, suitably lower power consumption printing should be used.

4. The motors required in the Artcam are all low
30 power miniature motors, and are typically only activated for a few seconds per photo.

5. The flash unit 45 is only used for some photographs. Its power consumption can readily be provided by a 3V lithium battery for a reasonably battery life.

35 6. The optional color display 5 is a major current drain for two reasons: it must be on for the whole time that

the camera is in use, and a backlight will be required if a liquid crystal display is used. Cameras which incorporate a color display will require a larger battery to achieve acceptable batter life.

5 Flash unit 56

 The flash unit 56 can be a standard miniature electronic flash for consumer cameras.

Artcam Central Processor 31

10 Turning now to Fig. 3, there is illustrated the Artcam central processor 31 in more detail. The ACP 31 can take many different forms depending on the technologies utilised. One form of ACP 31 is will now be described and includes the following components:

CPU Memory Decoder

15 The CPU Memory Decoder 68 is a simple decoder for satisfying CPU data accesses. The Decoder translates data addresses into DRAM addresses (which then get passed on to the Cache Interface) or into internal ACP register accesses over the internal low speed bus. The CPU Memory Decoder
20 allows for memory mapped I/O of ACP registers. A straightforward way of decoding is to use address bit 24. If bit 24 is clear, the address is in the lower 16 MB range, and hence can be directed to the Cache to be satisfied from DRAM. In most cases the DRAM will only be 8 MB, but we
25 allocate 16 MB to cater for a higher memory model Artcam. If bit 24 is set, the address represents an internal ACP register address. The address is translated into an access over the low speed bus to the requested component in the ACP.

30 Program Cache 77

 A small program cache is required for good performance. This requirement is mostly due to the use of a Rambus DRAM, which can provide high-speed data in bursts, but is inefficient for single byte accesses. 16 dedicated cache
35 lines of 32 bytes each will achieve most of the performance gain over no cache, and limits the cache size to 512 bytes.

The program cache gives increased performance for the CPU 72, and even allows small CPU functions to run completely from cache (and therefore simultaneously with VLIW processes). The Program Cache takes its data from the DRAM Memory Interface 81 or Program ROM interface 77. The data used by CPU programs comes through the CPU Memory Decoder and if the address is in DRAM, through the general Cache 176.

Cache Usage

10 The ACP contains a dedicated CPU instruction cache 77 and a general data cache 176.

 The data Cache 176 handles memory access requests that are have address bit 24 clear. If bit 24 is clear, the address is in the lower 16 MB range, and hence can be
15 satisfied from DRAM and the Data Cache. In most cases the DRAM will only be 8 MB, but we allocate 16 MB to cater for a higher memory model Artcam. If bit 24 is set, the address represents an internal ACP register address and is ignored by the Cache.

20 In order to reduce effective memory latency, the ACP contains 128 cache lines. Each cache line is 32 bytes wide. Thus the total amount of data cache is 4096 bytes (4k). Each cache line has a 4 bit group number associated with it, thereby allowing the splitting of the caches into 16
25 different groups. The caching groups must be contiguous sets of cache lines.

 All processor data requests use cache request group 0, and although the CPU can assign any number of cache lines (except none) to cache group 0, a minimum of 16 cache lines
30 is recommended for good performance.

 The other users of the cache interface - namely the Artcard Interface, the Display Controller, and the VLIW Vector Processor must use cache request groups appropriately. The CPU is responsible for ensuring that a
35 correct number of cache lines is assigned to each cache group for a given process. In any given cycle, 4

simultaneous accesses of 32 bits (4 bytes) to the caches are permitted. Each access must be to a separate group of cache lines.

JTAG Interface 85

5 A standard JTAG (Joint Test Action Group) Interface is included in the ACP for testing purposes. Due to the complexity of the chip, a variety of testing techniques are required, including BIST (Built In Self Test) and functional block isolation. An overhead of 10% in chip area is assumed
10 for overall chip testing circuitry.

Serial Interfaces

USB serial port interface 52

 This is a standard USB serial port, which is connected to the internal chip low speed bus, thereby allowing the CPU
15 to control it.

Keyboard interface 55

 This is a standard low-speed serial port, which is connected to the internal chip low speed bus, thereby allowing the CPU to control it. It is designed to be
20 optionally connected to a keyboard to allow simple data input to customize prints. Alternatives such as standard Infra-red Keyboard Interfaces could be provided.

Authentication chip serial interfaces 64

 These are 2 standard low-speed serial ports, which are
25 connected to the internal chip low speed bus, thereby allowing the CPU to control them.

 The reason for having 2 ports is to connect to both the on-camera Authentication chip 54, and to the print-roll Authentication chip 53 using separate lines. Only using 1
30 line may make it possible for a clone print-roll manufacturer to design a chip which, instead of generating an authentication code, tricks the camera into using the code generated by the authentication chip in the camera.

Parallel Interface 65

35 The parallel interface 65 connects the ACP to individual static electrical signals. The following is a

table of connections to the parallel interface:

Connection	Direction	Pins
Paper transport stepper motor	Output	4
Artcard stepper motor	Output	4
Zoom stepper motor	Output	4
Guillotine solenoid	Output	1
Flash trigger	Output	1
Status LCD segment drivers	Output	7
Status LCD common drivers	Output	4
Artcard illumination LED	Output	1
Artcard status LED (red/green)	Input	2
Artcard sensor	Input	1
Paper pull sensor	Input	1
Orientation sensor	Input	2
Buttons	Input	4
Total		36

5 The CPU is able to control each of these connections as
memory mapped I/O via the low speed bus.

DRAM Interface 81

Images are manipulated within the Artcam in a variety
of ways. Some methods of manipulation require random access
to pixels within an image, while others require access to
10 pixels in a specific logical order. The DRAM Interface
provides an interface between a client and the cached DRAM,
allowing specific known processing orders to be
appropriately cached.

15 The DRAM interface 81 includes 128 cached lines, each
32 bytes wide (32 bytes being the standard Rambus data
transfer unit).

The total memory on chip for caches is therefore 4096
bytes (128 x 32 bytes). The break up of cache assignment is:

20 -16 to cache the CPU's program (so programs can run at
the same time as control ACP processes)

-16 to cache CPU program's data

-96 floating. These can be assigned to ALUs for particular functions, or assigned to CPU program or data as desired.

5 The 128 cache lines are divided into 8 groups of 16 for separate addressing in a given cycle, with appropriate multiplexing.

As stated previously, the DRAM Interface 81 is responsible for interfacing between other client portions of the ACP chip and the RAMBUS DRAM. In effect, each module
10 within the DRAM Interface is an address generator.

There are three logical types of images manipulated by the ACP. They are:

-CCD Image, which is the Input Image captured from the
15 CCD.

-Internal Image format - the Image format utilised internally by the Artcam device.

Print Image - the Output Image format printed by the Artcam

20 These images are typically different in color space, resolution, and the output & input color spaces which can vary from camera to camera. For example, a CCD image on a low-end camera may be a different resolution, or have different color characteristics from that used in a high-end
25 camera. However all internal image formats are the same format in terms of color space across all cameras.

In addition, the three image types can vary with respect to which direction is 'up'. The physical orientation of the camera causes the notion of a portrait or landscape
30 image, and this must be maintained throughout processing. For this reason, the internal image is always oriented correctly, and rotation is performed on images obtained from the CCD and during the print operation.

CCD Image Organization

35 Although many different CCD image sensors could be utilised, it will be assumed that the CCD itself is a 750 x

500 image sensor, yielding 375,000 bytes (8 bits per pixel). Each 2x2 pixel block having the configuration as depicted in Fig.4.

5 A CCD Image as stored in DRAM has consecutive pixels with a given line contiguous in memory. Each line is stored one after the other. The image sensor Interface 83 is responsible for taking data from the CCD and storing it in the DRAM correctly oriented. Thus a CCD image with rotation 0 degrees has its first line G, R, G, R, G, R... and its
10 second line as B, G, B, G, B, G... If the CCD image should be portrait, rotated 90 degrees, the first line will be R, G, R, G, R, G and the second line G, B, G, B, G, B...etc.

15 Pixels are stored in an interleaved fashion since all color components are required in order to convert to the internal image format.

It should be noted that the ACP 31 makes no assumptions about the CCD pixel format, since the actual CCDs for imaging may vary from Artcam to Artcam, and over time. All processing that takes place via the hardware is controlled
20 by major microcode in an attempt to extend the usefulness of the ACP 31.

Internal Image Organization

Internal images typically consist of a number of channels. Vark images can include, but are not limited to:

25 Lab
Lab α
Lab Δ
 $\alpha\Delta$
L

30 L, a and b correspond to components of the Lab color space, α is a matte channel (used for compositing), and Δ is a bump-map channel (used during brushing, tiling and illuminating).

35 The VLIW processor 74 requires images to be organized in a planar configuration. Thus a Lab image would be stored

as 3 separate blocks of memory:

- one block for the L channel,
- one block for the a channel, and
- one block for the b channel

5 Within each channel block, pixels are stored contiguously for a given row (plus some optional padding bytes), and rows are stored one after the other.

Turning to Fig.5 there is illustrated an example form of storage of a logical image 100. The logical image 100 is
10 stored in a planar fashion having L 101, a 102 and b 103 color components stored one after another. Alternatively, the logical image 100 can be stored in a compressed format having an uncompressed L component 101 and compressed A and B components 105, 106.

15 Turning to Fig. 6, the pixels of for line n 110 are stored together before the pixels of for line and n + 1 (111). With the image being stored in contiguous memory within a single channel.

In the 8MB-memory model, the final Print Image after
20 all processing is finished, needs to be compressed in the chrominance channels. Compression of chrominance channels can be 4:1, causing an overall compression of 12:6, or 2:1.

Other than the final Print Image, images in the Artcam are typically not compressed. Because of memory constraints,
25 software may choose to compress the final Print Image in the chrominance channels by scaling each of these channels by 2:1. If this has been done, the PRINT Vark function call utilised to print an image must be told to treat the specified chrominance channels as compressed. The PRINT
30 function is the only function that knows how to deal with compressed chrominance, and even so, it only deals with a fixed 2:1 compression ratio.

Although it is possible to compress an image and then operate on the compressed image to create the final print
35 image, it is not recommended due to a loss in resolution. In addition, an image should only be compressed once - as the

final stage before printout. While one compression is virtually undetectable, multiple compressions may cause substantial image degradation.

Clip image Organization

5 Clip images stored on Artcards have no explicit support by the ACP 31. Software is responsible for taking any images from the current Artcard and organizing the data into a form known by the ACP. If images are stored compressed on an Artcard, software is responsible for decompressing them, as
10 there is no specific hardware support for decompression of Artcard images.

Image Pyramid Organization

 During brushing, tiling, and warping processes utilised to manipulate an image it is often necessary to compute the
15 average color of a particular area in an image. Rather than calculate the value for each area given, these functions make use of an image pyramid. As illustrated in Fig.7, an image pyramid is effectively a multi-resolution pixel-map. The original image 115 is a 1:1 representation. Low-pass
20 filtering and sub-sampling by 2:1 in each dimension produces an image $\frac{1}{4}$ the original size 116. This process continues until the entire image is represented by a single pixel. An image pyramid is constructed from an original internal format image, and consumes $\frac{1}{3}$ of the size taken up by the
25 original image ($\frac{1}{4} + \frac{1}{16} + \frac{1}{64} + \dots$). For an original image of 1500 x 1000 the corresponding image pyramid is approximately $\frac{1}{2}$ MB. An image pyramid is constructed by a specific Vark function, and is used as a parameter to other Vark functions.

30 Print Image Organization

 The entire processed image is required at the same time in order to print it. However the Print Image output can comprise a CMY dithered image and is only a transient image format, used within the Print Image functionality. However,
35 it should be noted that color conversion will need to take place from the internal color space to the print color

space. In addition, color conversion can be tuned to be different for different print rolls in the camera with different ink characteristics e.g. Sepia output can be accomplished by using a specific sepia toning Artcard, or by using a sepia tone print-roll (so all Artcards will work in sepia tone).

Color Spaces

As noted previously there are 3 color spaces used in the Artcam, corresponding to the different image types.

The ACP has no direct knowledge of specific color spaces. Instead, it relies on client color space conversion tables to convert between CCD, internal, and printer color spaces:

CCD:RGB

Internal:Lab

Printer:CMY

Removing the color space conversion from the ACP allows:

- Different CCDs to be used in different cameras

- Different inks (in different print rolls over time) to be used in the same camera

- Separation of CCD selection from ACP design path

- A well defined internal color space for accurate color processing

The overall process for creating an output image is as illustrated in Fig.8. The process includes reading in a CCD image in a CCD color space, the conversion of the CCD image to an internal image in an internal color space, the continual processing of the internal image to produce a final internal image, followed by the creation of a print image for printing out in the printer's color space. With each conversion, color tables are required for color mapping the images from one color space to another. These color tables can be provided in the Artcam ROM or in the particular print ROM.

DRAM Interface

The DRAM used by the Artcam is a 64Mbit (8MB) RAMBUS Dram operating at 500 Mbytes/sec. Using RAMBUS DRAM implies that applications should minimize the number of random memory accesses to avoid degraded memory access performance.

5 To take advantage of the 4 internal banks of memory in a single DRAM chip, every 32 bytes should be in a different bank with address wiring accordingly. The 4 bank internal arrangement of RAMBUS DRAM can also be used to advantage if necessary as long as this does not create unnecessary
10 algorithmic complexity.

Bank accesses can have their latencies overlapped, so while data is being transferred from one bank, another can be setting up for the transfer. Interleaved in this way, assuming a worst case of a DRAM-internal-cache miss every
15 access, 4 sets of 32 byte reads can be accomplished in 320ns.

Cache Lines 176

In order to reduce effective memory latency, the ACP contains 128 cache lines in Data Cache 176, each 32 bytes
20 wide. The total memory on chip for caches is therefore 4096 bytes (128 x 32 bytes). The break up of cache assignment is:

16 to cache the CPU's program (so programs can run at the same time as control ACP processes)
16 to cache CPU program's data
25 96 floating. These can be assigned to ALUs for particular functions, or assigned to CPU program or data as desired.

The 128 cache lines are divided into 8 groups of 16 for separate addressing in a given cycle, with appropriate
30 multiplexing.

Memory Organization

Memory in an Artcam consists of a contiguous 32MB area (of which 8 MB is actually used). In addition to the real memory, there are some other non-contiguous address spaces
35 which are effectively 'virtual' memory areas. These are ACP registers, used for memory mapped I/O. The memory

organization for an Artcam with 8MB of RDRAM is shown in the following table:

Program scratch RAM	0.50 MB
Artcard data	1.00 MB
Photo Image, captured from CCD	0.50 MB
Print Image (compressed)	2.25 MB
1 Channel of expanded Photo Image	1.50 MB
1 Image Pyramid of single channel	1.00 MB
Intermediate Image Processing	1.25 MB
TOTAL	8 MB

5 Uncompressed, the Print Image requires 4.5MB (1.5MB per
channel). To accommodate other objects in an 8MB model, the
Print Image needs to be compressed. If the chrominance
channels are compressed by 4:1 they require only 0.375MB
each). The memory model described here assumes a single 8
10 MB RDRAM. Other models of the Artcam may have more memory,
and thus not require compression of the Print Image. In
addition, with more memory a larger part of the final image
can be worked on at once, potentially giving a speed
improvement. The ejecting or inserting an Artcard
15 invalidates the 5.5MB area holding the Print Image, 1
channel of expanded photo image, and the image pyramid. This
space may be safely used by the Artcard Interface for
decoding the Artcard data.

Program Memory^{*} 70 (Fig. 2)

20 A program memory 70 is provided for storing Artcam start up
programs, color conversion tables and other necessary data.
The program memory ROM interacts with the ACP 31 via the
Program ROM interface 71 (Fig. 3).

VLIW Input and Output FIFOs

25 The VLIW Input 75 and Output 76 FIFOs are 8 bit wide
FIFOs used for communicating between processes and the VLIW
Vector Processor 74. Both FIFOs 75, 76 are under the control
of the VLIW Vector Processor 74 , but can be cleared and

queried (e.g. for status) etc. by the CPU 72.

VLIW Input FIFO 75

A client writes 8 bit data to the VLIW Input FIFO 75 in order to have the data processed by the VLIW Vector Processor 74. Clients include the Image sensor Interface 83, the Artcard Interface 87, and CPU 72. Each of these processes is able to offload processing by simply writing the data to the FIFO 75, and letting the VLIW Vector Processor do the work.

An example of the use of a client's use of the VLIW Input FIFO 75 is the Image sensor Interface (ISI) 83. The ASI takes data from the CCD and writes it to the FIFO 75. A VLIW process takes it from there, transforming it into the correct image data format, and writing it out to DRAM. The ASI becomes much simpler as a result.

VLIW Output FIFO 76

A client reads 8 bit data from the VLIW Output FIFO 76 after it has been processed by the VLIW Vector Processor 74. Clients can include the Print Head Interface 62 and the CPU 72. Each of these processes is able to offload processing by simply reading the already processed data from the FIFO 76.

An example of the use of a client's use of the VLIW Output FIFO 76 is the Print Head Interface (PHI) 62. A VLIW process takes an image, rotates it to the correct orientation, color converts it, and dithers the resulting image according to the print head requirements. The PHI 62 reads the dithered formatted 8 bit data and simply passes it on to the Print Head external to the ACP. The PHI becomes much simpler as a result.

VLIW Vector Processor 74

To achieve the high processing requirements of Artcam, the ACP contains the VLIW (Very Long Instruction Word) Vector Processor 74. The VLIW processor is illustrated in detail in Fig. 18 and is a set of 8 identical ALU units e.g. 178 working in parallel, connected by a crossbar switch 183. Each ALU 178 can perform four 8 bit multiplications, eight 8

bit additions, three 32 bit additions, I/O processing, and various logical operations in each cycle. The ALUs are microcoded, and each has an address generator to allow full use of available cycles for data processing. The eight ALUs
5 are normally synchronized to provide a tightly interacting VLIW processor.

Clocking at 100 MHz, the VLIW Vector Processor runs at 12 Gops (12 billion operations per second). Instructions are tuned for image processing functions such as warping,
10 artistic brushing, complex synthetic illumination, color transforms, image filtering, and compositing.

The VLIW Vector Processor 74 is 8 ALUs connected to form a ring, with data from each ALU e.g. 178 available to its immediate left 184 and right 185 neighbors. The
15 crossbar switch 183 connects all ALUs such that each ALU can provide one input to, and takes two outputs from, the crossbar switch. Two common registers form a control and synchronization mechanism for the ALUs.

The data bus 182 comprises 4 Cache buses which allow
20 connectivity to DRAM via the Cache 176. Each of the buses can be used in a single cycle, as long as the accesses are to a different cache group.

Turning to Fig. 19, there is illustrated a single ALU unit eg. 178. Each ALU unit 178 contains two FIFOs 186, 187
25 for transferring 16 bit data between the ALU unit and the DRAM (via the Cache 176). Data taken from the Cache is transferred to an ALU proper 188 via the Input FIFO 186. Output data is written to the Output FIFO 187 and from there to the Cache. These two FIFOs are different from the VLIW
30 Input FIFO 75 and VLIW Output FIFO 76. Firstly, they are 16 bit instead of 8 bit. Secondly, the FIFOs are internal to the VLIW Vector processor 74 (one pair per ALU 187), while all ALUs share a single VLIW Input FIFO and VLIW Output FIFO.

35 The Address Generator 189 is an I/O Unit, controlling data flow between the Cache (via the 4 cache buses 186) and

the ALU proper 188 (via the Input and Output FIFOs 186,187). The address generator 189 is able to read and write data (specifically images in a variety of formats) as well as tables and simulated FIFOs in DRAM. The formats are
5 customizable under software control.

The ALU proper 188 consists of a number of registers, some arithmetic logic for processing data, microcode RAM, and connections to the outside world (including other ALUs). It is controlled by a local ALU state machine 194 that runs
10 the client's microcode.

A low speed control data bus 195 (Fig. 18) allows the CPU to read and write registers in the ALU, update microcode, as well as the common registers shared by all ALUs in the VLIW Vector Processor

15 Microcode Ram 196

Each of the 8 ALUs contains a microcode RAM 196 to hold the program for that particular ALU. Rather than have the microcode in ROM, the microcode is in RAM, with the program CPU responsible for loading it up. For the same space on
20 chip, this tradeoff reduces the maximum size of any one function to the size of the RAM, but allows an unlimited number of functions to be written in microcode. Functions implemented using ALU microcode can include Vark acceleration, Artcard reading, and Printing.

25 The VLIW Vector Processor scheme has several advantages for the case of the ACP:

Hardware design complexity is reduced

Hardware risk is reduced due to reduction in complexity

Hardware design time does not depend on all Vark

30 functionality being implemented in dedicated silicon.

Space on the chip is reduced overall (due to large number of processes able to be implemented as microcode)

Functionality can be added to Vark (via microcode) with
35 no impact on hardware design time.

Size and Content

The CPU loaded microcode RAM 196 for controlling each ALU is 48 words, with each word being 103 bits wide. A summary of the microcode size for control of various sections of the ALU is listed in the following table:

Process Block	Size (bits)
Status Output	3
Branching (microcode control)	11
In ₁	4
In ₂	4
In ₃	2
In ₄	2
Out ₁	3
Out ₂	3
Read	9
Write	5
Latches ₁	10
Latches ₂	5
Adder/Logical	14
Multiply/Interpolate	16
Barrel Shifter	12
TOTAL	103

5

With 48 instruction words, the total microcode RAM per ALU is 4944 bits. With 8 identical ALUs this equates to 4944 bytes, less than 5K in total. Some of the bits of the microcode word are actual control bits, while others are decoded. The meanings of each of the bits of the microcode word being further described hereinafter.

10

Synchronization Between ALUs

15

Each ALU contains an 8 bit Synchronization Register 197. It is a mask used to determine which ALUs 186 work together, and has one bit set for each of the corresponding ALUs that are functioning together. For example, if all of the ALUs were functioning together each of the 8 Synchronization registers would have all 8 bits set. If

there were two asynchronous processes of 4 ALUs each, four of the ALUs would have 4 bits set in their Synchronization Registers (corresponding to themselves), and the other four would have the other 4 bits set in their Synchronization
5 Registers (corresponding to themselves).

The Synchronization Register is used in two basic ways:

Stopping and starting a given process in synchrony

Suspending execution within a process

Stopping and Starting Processes

10 The CPU is responsible for loading the microcode RAM 196 and loading the execution address for the first instruction (usually 0).

Execution of microcode only occurs when all the bits of the Synchronization Register are also set in the Common
15 Synchronization Register 197. The CPU can therefore set up all the ALUs and then start or stop processes with a single write to the Common Synchronization Register 197. This synchronization scheme allows multiple processes to be running asynchronously on the ALUs 186, being stopped and
20 started as processes rather than one ALU at a time.

Suspending Execution within a Process

When any of the ALUs attempts to read from or write to a VLIW input or output FIFO (based on the opcode of the current microcode instruction), and is unable to (for
25 example the FIFO is empty on a read request, or full on a write request), that ALU will assert a SuspendProcess control signal, 198. The SuspendProcess signals from all 8 ALUs are fed to all the 8 ALUs. The Synchronization Register 197 is ANDed with the 8 input SuspendProcess values, and if
30 the result is non-zero, none of the register WriteEnables will be set. The practical upshot is that no registers or FIFOs will be updated during that cycle by the ALUs that form the same process group as the ALU that was unable to complete its task.

35 Each cycle the ALU state machine 194 will attempt to re-execute the microcode 196, and will continue to do so

until successful. Of course the Common Synchronization Register can be written to by the CPU to stop the entire process if necessary. This synchronization scheme allows any combinations of ALUs to work together, each group only affecting its co-workers with regards to suspension due to data not being ready for reading or writing.

Control and Branching

Each ALU proper 188 contains four basic input and calculation units (Read, Adder/Logic, Multiply/Interpolate, and Barrel Shifter). During each cycle, each of the four basic input and calculation units produces two status bits: a Zero flag and a Negative flag indicating whether the result of the operation during that cycle was 0 or negative. Each cycle one of those 8 status bits is chosen by microcode instructions to be output 199 from the ALU proper. Each of the 8 ALUs outputs 1 status bit, all of which are combined into an 8 bit Common Status Register 200. During the next cycle, each ALU's microcode program can select one of the 8 bits from the Common Status Register 200, and branch to another microcode address dependent on the value of the status bit.

Status bit

The output Status bit from a given ALU is one of the 8 status bits from the various input and calculation units. Only one of the 4 processor block units will assert the value on the 1 bit tri-state status bit bus. The status bit is output to be combined with the other ALU status bits to update the Common Status Register.

The microcode for determining the output status bit takes the following form:

# Bits	Description
2	Select processor block whose status bit is to be output
1	0 = Zero flag 1 = Negative flag

3	TOTAL
---	-------

Within the ALU, the 2 bit Select Processor Block bits are decoded into four 1 bit enable bits, with a different enable bit sent to each processor block. The status select bit (Zero or Neg) is passed into all processor blocks to determine which bit is to be output onto the status bit bus.

Branching Within Microcode

Each cycle, a microcode program has the ability to branch to another address based on the value of a status bit from the Common Status Register. If a branch is not taken from microcode address N, the next microcode address to be executed will be N+1. The microcode for determining the next execution address takes the following form:

# Bits	Description
2	00 = NOP (next address = current+1) 01 = Jump always 10 = Jump if status bit clear 11 = Jump if status bit set
3	Select status bit from status word
6	Address to jump to (absolute address, choice of 48 only)
11	TOTAL

ALU 188

Turning now to Fig. 20, there is show the ALU proper 188 in more detail. Inside each ALU proper 188 are a number of specialized processing blocks, controlled by a microcode program. The specialized processing blocks include:

Read block 202, for accepting data from the input FIFOs
Write block 203, for sending data out via the output FIFOs

Adder/Logical block 204, for addition & subtraction, comparisons and logical operations

Multiply/Interpolate block 205, for multiple types of interpolations and multiply/accumulates

BarrelShift block 206, for shifting data as required
Input blocks eg. 207, for accepting data either from
the external crossbar switch or the neighboring ALUs
Output blocks, for sending data either to the external
5 crossbar switch or the neighboring ALUs

Four specialized 32 bit registers hold the results of the 4
main processing blocks:

M register 209 holds the result of the
Multiply/Interpolate block 205
10 L register 210 holds the result of the Adder/Logic
block 204
S register 211 holds the result of the Barrel Shifter
block 206

R register 212 holds the result of the Read block 202
15 In addition there are internal crossbar switches 213,
214 for data transport, registers for temporary
storage/results, and some control logic.

The various process blocks are further expanded in the
following sections, together with the microcode definitions
20 that pertain to each block. The microcode is decoded within
a block to provide the control signals to the various units
within.

Data Transfers Between ALUs

Each ALU unit is able to exchange data via the external
25 crossbar and also with its left and right neighbors
directly. Each ALU unit has a single output to the external
crossbar, but two inputs from the external crossbar. In this
way two operands for processing can be read in a single
cycle, but cannot be actually used in an operation until the
30 following cycle. Data transferred from neighboring ALUs
however, can be used within the same cycle.

Thus there are 4 input process blocks and 2 output
process blocks. The 4 input process blocks eg. 207 deal with
2 data inputs from the external crossbar as well as the left
35 and right neighboring ALUs. The 2 output process blocks deal
with outputting data to the crossbar, and to the neighboring

ALUs (the same output is sent to both neighboring ALUs).

In₁ and In₂

- Each of these two identical process blocks accepts input from the crossbar switch during a given cycle. The data is stored locally in a latch e.g. 207. The selection bits for choosing from among the 8 inputs are output to the external crossbar switch. The microcode for both In₁ and In₂ takes the same form:

# Bits	Description
1	0 = NOP 1 = Load data from crossbar
3	Select input from crossbar (8 choices, 1 from each ALU)
4	TOTAL

In₃ and In₄

- Each of these two identical process blocks accepts data from a neighboring ALU during a given cycle. The data can be stored locally in a latch, transferred directly to another process, or both. Each of the two blocks has the same structure as illustrated in Fig. 21. The microcode for both In₃ and In₄ takes the same form:

# Bits	Description
1	0 = NOP 1 = Load data from Left ALU
1	0 = Output = Left ALU 1 = Output = local data
2	TOTAL

Out₁

- Complementing In₁ and In₂ (which accept data from the crossbar) is Out₁, data output to the crossbar for selection by other ALUs (via their In₁ and In₂ process blocks). The data is chosen from any of the inputs to the Output Crossbar 214 (results of the previous ALU operations or either of the 2 output registers from the Regs₂ block). The microcode for Out₁ takes the following form:

# Bits	Description
3	Select Out ₁ from the Output Crossbar
3	TOTAL

The output sent to the crossbar is the same as that sent to the Write sub-process. This means that in the same cycle different data is not available to other ALUs except as neighboring data via Out₂. Typically there is no need to pass data on when a FIFO is being written to (via the Write sub-process).

Out₂

Complementing In₃ and In₄ (which accept data from neighboring ALUs) is Out₂, data output to neighboring ALUs. The same data is sent to both neighboring ALUs. The data is chosen from any of the inputs to the Output Crossbar (results of the previous ALU operations or either of the 2 output registers from the Regs₂ block). The microcode for Out₂ takes the following form:

# Bits	Description
3	Select Out ₂ from the Output Crossbar
3	TOTAL

Latches₂ 215

Sometimes it is necessary to delay an output for several cycles, or to send packets of data a portion at a time. The register set Latches₂ 215 contains two 32 bit registers to hold temporary variables for output. In each cycle either of the registers can be updated from the Internal Crossbar 213, and both registers are made available to the Output Crossbar 214 for selection as Out₁/Write or Out₂. The CPU has no direct access to these registers. The Latches₂ sub-process is as illustrated in Fig. 22. The microcode for Latches₂ takes the following form:

# Bits	Description
3	Select data from Internal Crossbar

1	0 = NOP 1 = Load Latch ₁ from Internal crossbar
1	0 = NOP 1 = Load Latch ₂ from Internal crossbar
5	TOTAL

Data Transfers Between ALUs and DRAM or External Processes

As shown in Fig. 19, for transferring data between an ALU 188 and DRAM, a local Read FIFO 186 and a local Write FIFO 187 is connected to each ALU 188. For transferring data between all ALUs and external processes there is one common VLIW Input FIFO 75 and one common VLIW Output FIFO 76 to which all ALUs have access. The VLIW Input and Output FIFOs are only 8 bits wide, and are used for printing, Artcard reading, transferring data to the CPU etc. The local Input and Output FIFOs are 16 bits wide. The local Input and local Output FIFOs are connected to the I/O section of each the 8 VLIW Process Units, and communicate with the DRAM via the Cache.

Read Register 202 (Fig. 20)

The Read process block 202 is responsible for updating the ALU's R register 212, which represents the external input data to a VLIW microcoded process. Each cycle the Read Unit 202 is able to read from either the common VLIW Input FIFO (8 bits) or the local Input FIFO (16 bits). A 32 bit value is generated, and then all or part of that data is transferred to the R register 212. The process is illustrated in Fig. 23.

The microcode for the Read register is described in the following table. The interpretations of some bit patterns can be deliberately chosen to aid decoding.

9.# Bits	Description
2	00 = load with 0 (no read) 01 = Read 8 bits from VLIW Input FIFO 10 = Read 16 bits from Local FIFO (pad with 0)

	or sign extend) 11 = Read 8 bits from Local FIFO (pad with 0 or sign extend)
1	0 = Treat data as unsigned (pad with 0) 1 = Treat data as signed (sign extend when reading from FIFO) r
2	How much to shift data left by: 00 = 0 bits (no change) 01 = 8 bits 10 = 16 bits 11 = 24 bits
4	Which bytes of R to update Each of the 4 bits represents 1 byte WriteEnable on R
9	TOTAL

Write Register 203

The Write process block is able to write to either the common VLIW Output FIFO or the local Output FIFO each cycle.

- 5 The data value selected as Out₁ is the input to the Write process block. In addition, since both FIFOs are not written to at the same time, only one 16 bit value is output to both FIFOs, with the low 8 bits going to the VLIW Output FIFO. The microcode controls which of the two FIFOs gates in the
- 10 value. The value Out₁ is determined from the Output Crossbar. The microcode for Write takes the following form:

# Bits	Description
2	00 = NOP 01 = Write VLIW Output FIFO 10 = Write local Output FIFO 11 = Reserved
3	Select part of Out ₁ to write (32 bits = 4 bytes ABCD) 000 = 0D 001 = 0D

	010 = 0B 011 = 0A 100 = CD 101 = BC 110 = AB 111 = 0
5	TOTAL

Computational Blocks

Each ALU has two computational process blocks, namely an Adder/Logic process block 204, and a Multiply/Interpolate process block 205. In addition there are some other blocks that provide help to these computational blocks - a Barrel Shifter process block 206 and a set of registers that permits intermediate storage and delays in pipelined operations.

10 Barrel Shifter 206

The Barrel Shifter process block takes its input from the output of Adder/Logic or Multiply/Interpolate process blocks or the previous cycle's results from those blocks (ALU registers L and M). The 32 bits selected are barrel shifted an arbitrary number of bits in either direction (with sign extension as necessary), and output to the ALU's S register. The process can be as illustrated in Fig. 24. The microcode for the Barrel Shift process block is described in the following table.

# Bits	Description
3	000 = NOP 001 = Shift Left (unsigned) 010 = Reserved 011 = Shift Left (signed) 100 = Shift right (unsigned, no rounding) 101 = Shift right (unsigned, with rounding)

	110 = Shift right (signed, no rounding) 111 = Shift right (signed, with rounding)
2	Select Input to barrel shift: 00 = Multiply/Interpolate result 01 = M 10 = Adder/Logic result 11 = L
5	# bits to shift
1	Ceiling of 255
1	Floor of 0 (signed data)
12	TOTAL

Adder/Logic 204

The Adder/Logic process block is designed for simple 32 bit addition/subtraction, comparisons, and logical operations.

- 5 In a single cycle a single addition, comparison, or logical operation can be performed, with the result stored in the ALU's L register. There are two primary operands, A and B, which are selected via the Internal Crossbar 213. The crossbar selection allows the results of the previous
- 10 cycle's arithmetic operation to be used or an input from another ALU. Operands A and B may also be chosen from Reg₁ Reg₂ of Latch1 and two constants, internal to the Adder/Logic process block. The internal structure of the Adder/Logic process is as illustrated in Fig. 25.

- 15 The microcode for the Adder/Logic process block is described in the following table

#Bits	Description
4	0000 = A+B (carry in = 0) 0001 = A+B (carry in = carry out of previous operation) 0010 = A+B+1 (carry in = 1) 0011 = A+1 (increments A) 0100 = A-B-1 (carry in = 0)

	<p>0101 = A-B (carry in = carry out of previous operation)</p> <p>0110 = A-B (carry in = 1)</p> <p>0111 = A-1 (decrements A)</p> <p>1000 = NOP</p> <p>1001 = ABS (A-B)</p> <p>1010 = MIN(A, B)</p> <p>1011 = MAX(A, B)</p> <p>1100 = A AND B (both A & B can be inverted, see below)</p> <p>1101 = A OR B (both A & B can be inverted, see below)</p> <p>1110 = A XOR B (both A & B can be inverted, see below)</p> <p>1111 = A (A can be inverted, see below)</p>
1	<p>If logical operation:</p> <p>0 = A=A</p> <p>1 = A=NOT (A)</p> <p>If Adder operation:</p> <p>0 = A is unsigned</p> <p>1 = A is signed</p>
1	<p>If logical operation:</p> <p>0 = B=B</p> <p>1 = B=NOT (B)</p> <p>If Adder operation</p> <p>0 = B is unsigned</p> <p>1 = B is signed</p>
4	Select A [Internal Crossbar (8), Reg ₁ , Reg ₂ , K ₁ , K ₂]
4	Select B [Internal Crossbar (8), Reg ₁ , Reg ₂ , K ₁ , K ₂]
14	TOTAL

Latches₁ 216

- Sometimes it is necessary to delay an operation for one or more cycles. The register set Latches₁ 216 contains four 32 bit registers to hold temporary variables during processing.
- Each cycle one of the registers can be updated from the Internal Crossbar 213, and two registers are made available to other process blocks (namely the Adder/Logic process block 204 and the Multiply/Interpolate process block 205).
- The CPU has direct access to these registers. The structure of the Latches₁ sub-process is as illustrates in Fig. 26.
- The microcode for Latches₁ takes the following form:

# Bits	Description
1	0 = NOP 1 = Load latch from Internal crossbar
2	Which latch to load (1 of 4)
3	Select data to load from Internal crossbar
2	Select Reg ₁ from the 4 latches
2	Select Reg ₂ from the 4 latches
10	TOTAL

Multiply/Interpolate 205

- The Multiply/Interpolate process block 205 is a set of four 8 x 8 interpolator units that are capable of performing four individual 8 x 8 interpolates per cycle, or can be combined to perform a single 16 x 16 multiply. This gives the possibility to perform up to 4 linear interpolations, a single bi-linear interpolation, or half of a tri-linear interpolation in a single cycle. The result of the interpolations or multiplication is stored in the ALU's L register. There are two primary operands, A and B, which are selected via the Internal Crossbar. The crossbar selection allows the results of the previous cycle's arithmetic operation to be used or an input from another ALU. Operands A and B may also be chosen from Reg₁, Reg₂, or two constants, internal to the Multiply/Interpolate process

block. The structure of the multiply/interpolate block can be as indicated in Fig. 27.

Each interpolator block functions as a simple 8 bit interpolator [result = $A + (B-A)f$] or as a simple 8 x 8 multiply [result = $A * B$]. When the operation is
5 multiply [result = $A * B$]. When the operation is interpolation, A and B are treated as four 8 bit numbers A_0 through A_3 (A_0 is the low order byte), and B_0 through B_3 .

10 Agen, Bgen, and Fgen are responsible for ordering the inputs to the interpolate units so that they match the operation being performed. For example, to perform bilinear interpolation, each of the 4 values must be multiplied by a different factor and the result summed, while a 16 x 16 bit multiplication requires the factors to be 0.

15 The microcode for the Adder/Logic process block is described in the following table.

# Bits	Description
4	$0000 = (A_{10} * B_{10}) + V$ $0001 = (A_0 * B_0) + (A_1 * B_1) + V$ $0010 = (A_{10} * B_{10}) - V$ $0011 = V - (A_{10} * B_{10})$ $0100 = \text{Interpolate } A_0, B_0 \text{ by } f_0$ $0101 = \text{Interpolate } A_0, B_0 \text{ by } f_0, A_1, B_1 \text{ by } f_1$ $0110 = \text{Interpolate } A_0, B_0 \text{ by } f_0, A_1, B_1 \text{ by } f_1, A_2, B_2 \text{ by } f_2$ $0111 = \text{Interpolate } A_0, B_0 \text{ by } f_0, A_1, B_1 \text{ by } f_1, A_2, B_2 \text{ by } f_2, A_3, B_3 \text{ by } f_3$ $1000 = \text{Interpolate 16 bits stage 1 } [M = A_{10} * f_{10}]$ $1001 = \text{Interpolate 16 bits stage 2 } [M = M + (A_{10} * f_{10})]$ $1010 = \text{Tri-linear interpolate A by f stage 1 } [M = A_0 f_0 + A_1 f_1 + A_2 f_2 + A_3 f_3]$ $1011 = \text{Tri-linear interpolate A by f stage 2 } [M = M + A_0 f_0 + A_1 f_1 + A_2 f_2 + A_3 f_3]$

	1100 = Bi-linear interpolate A by f stage 1 $[M=A_0f_0+A_1f_1]$ 1101 = Bi-linear interpolate A by f stage 2 $[M=M+A_0f_0+A_1f_1]$ 1110 = Bi-linear interpolate A by f complete $[M=A_0f_0+A_1f_1+A_2f_2+A_3f_3]$ 1111 = NOP
4	Select A [Internal crossbar (8), Regs ₁ (2), K ₁ , K ₂]
4	Select B [Internal Crossbar (8), Regs ₁ (2), K ₁ , K ₂]
If Mult:	
2	Select V [Adder result, M, 0]
1	Treat A as signed
1	Treat B as signed
If Interp :	
3	Select basis for f [Internal Crossbar (8), Regs ₁ (2), K ₁ , K ₂]
1	Select interpolation f generation from P ₁ or P ₂ P _n is interpreted as # fractional bits in f If P _n =0, f is range 0..255 representing 0..1
16	TOTAL

I/O Address Generator 189

Returning to Fig. 19, a VLIW process does not access DRAM directly. Instead, a specific ALU reads data from its local Input FIFO 186, and writes data to its local Output FIFO 187. The I/O Address Generator 189 is responsible for reading the required data from DRAM and placing it into the Input FIFO, and is responsible for taking the data from the Output FIFO and writing it to DRAM.

The I/O Address Generator is a state machine responsible for generating addresses and control for data retrieval and storage in DRAM via the Cache. It is

customizable under CPU software control. The address generator produces addresses in two broad categories:

- Image Iterators, used to iterate (reading, writing or both) through pixels of an image in a variety of ways
- 5 ◦ Table I/O, used to randomly access pixels in images, data in tables, and to simulate FIFOs in DRAM

The I/O Address Generator 189 is connected to the 4 Cache Interface buses 186. Of the 8 ALUs, four simultaneous 16 bit accesses by the VLIW processor to the Cache can occur
10 in a single cycle.

The various types of address generation (Image Iterators and Table I/O) have been previously described and are further described in the subsequent sections.

Registers

15 The I/O Address Generator 189 has a set of registers that are used to control address generation. The addressing mode also determines how the data is formatted and sent into the local Input FIFO, and how data is interpreted from the local Output FIFO. The CPU is able to access the registers
20 of the I/O Address Generator via the low speed bus.

Caching

Several registers are used to control the caching mechanism, specifying which cache group to use for inputs, outputs etc.

Register Name	# bits	Description
ReadCacheGroup	4	Which cache group to read data from
ReadCacheBus	2	Which bus to use to access Cache Interface for Reads
WriteCacheGroup	4	Which cache group to write data to
WriteCacheBus	2	Which bus to use to access Cache Interface for writes

25

Defining the Characteristics of the Data

Image Iterators and Table I/O both address DRAM according to specifications defined by a set of common registers. Although the characteristics are defined in terms of an image (with a width and a height etc.), the registers are generic to any organized data block. These registers are defined in the following table:

Register Name	# bit s	Description
ImageStart	32	The address in memory where the image starts
ImageHeight	12	The number of lines in the image
ImageWidth	12	The number of pixels in a line
PixelOffset	12	The number of bytes from one entry to the next
RowOffset	12	The number of bytes from one row to the next. Equals ImageWidth + any padding
StartRow	12	Which row to start at in the image
EndRow	12	The last row+1 to be returned or written to within the image
StartPixel	12	Left border of the section of the image
EndPixel	12	The last pixel+1 to be returned or written to along a given row.
ReadEnable	1	Reads the image according to the Iterator type
WriteEnable	1	Writes the data according to the Iterator type
ConstantPixel	16	Constant pixel value to return if read is out of bounds and DuplicateEdge =0.
DuplicateEdge	1	0 = return edge value for out of bounds reads

		1 = return ConstantPixel for out of bounds reads
Address Mode	4	0000 - 0111 = Image Iterator 1000 - 1111 = Table I/O
AccessSpecific ₁	16	the specific addressing mode determines the usage
AccessSpecific ₂	16	the specific addressing mode determines the usage
AccessSpecific ₃	16	the specific addressing mode determines the usage
AccessSpecific ₄	16	the specific addressing mode determines the usage

5 All writes out of the image bounds are simply ignored, although the appropriate internal address counters are incremented. Writing out of bonds is useful for starting up and terminating processes. To write out of bounds to the left of an image, the StartPixel value should be less than 0, and to write out of bounds to the right of an image, the EndPixel should be greater than the ImageWidth.

10 It is possible to point into a sub-section of an image, by setting ImageStart at the top left corner, and ImageWidth and ImageHeight to be the width and height respectively of the section. Advancement from one line of an image to the next is accomplished using RowOffset, to enable access to a portion of an image as if it were an image. The appropriate
15 bonding boxes should be used (eg 32 byte boundaries) in order to minimize algorithmic complexity.

As noted previously, the primary image pixel access method for software and hardware algorithms is via Image Iterators located within the I/O address generator 189.
20 Image iterators perform all of the addressing and access to the caches of the pixels within an image channel and read, write or read & write pixels for their client. Read Iterators read pixels in a specific order for their clients, and Write Iterators write pixels in a specific order for

their clients. Clients of Iterators read pixels from the local Input FIFO or write pixels via the local Output FIFO.

Read Image Iterators read through an image in a specific order, placing the pixel data into the local Input
5 FIFO 186. Every time a client reads a pixel from the Input FIFO, the Read Iterator places the next pixel from the image (via the Cache Interface) into the FIFO.

Write Image Iterators write pixels in a specific order to write out the entire image. Clients write pixels to the
10 Output FIFO that is in turn read by the Write Image Iterator and written to DRAM via the Cache.

Typically a VLIW process will have its input tied to a Read Iterator, and output tied to a corresponding Write
15 Iterator. From the client's perspective, the FIFO is the effective interface to DRAM. The actual method of carrying out the storage (apart from the logical ordering of the data) is not of concern. Although the FIFO is perceived to be effectively unlimited in length, in practice the FIFO is of limited length, and there can be delays storing and
20 retrieving data, especially if several memory accesses are competing. The 4 bit Address Mode Register is used to determine the Iterator type:

Bit #	Address Mode
3	0 = This addressing mode is an Iterator
2	0 = Do not loop through data 1 = Loop through data
1 and 0	Iterator Mode 00 = Sequential Iterator 01 = Box Iterator [read only] 10 = Vertical Strip Iterator 11 = Reserved

Image access

25 Access to images is via special image address generators, defined logically below. As will become more apparent hereinafter the VLIW Vector processor 74 contains a

number of these address generation state machines (AGSM).

Random Access to pixels

Images are rarely required to be accessed in completely random (x, y) fashion, although it is straightforward enough
5 to access a given pixel within a channel by the following addressing algorithm:

Address for pixel (X, Y) = ImageStart + (RowOffset * Y) + X.

This only gives the address of a single color channel's component, and 3 such operations would be required to access
10 all 3 color components of a single pixel.

Image Iterators allow Sequential Access to pixels

The primary image pixel access method for software and hardware algorithms is via Image Iterators located within the VLIW Vector Processor 74. Image iterators perform all
15 of the addressing and caching of the pixels within an image channel and either read or write pixels for their client. Read Iterators read pixels in a specific order for their clients, and Write Iterators write pixels in a specific order for their clients.

Turning to Fig.9, there is illustrated the operation of the Image Iterators of the embodiment. The Read Iterator, , and Write Iterators act as an intermediary between a client requesting the data and the data stored within the DRAM 33. The iterators are responsible for correct ordering of image
20 data.
25

Turning to Fig.10, there is a illustrated an example Read Iterator 130 which can comprise a state machine 136 interconnected and controlling a FIFO 137. The state machine 136 is responsible for sending the requests to the DRAM and keeping the FIFO 137 full. Further, the state
30 machine 136 receives read requests from clients and clocks-out FIFO data from the FIFO queue 137 in response to those read requests.

The Read Image Iterators 130 can be thought of as a
35 FIFO that contains the entire image in a specific order (of course they are not implemented as such). Every time a pixel

is read from the FIFO 137, the next pixel from the image is read into the end of the FIFO.

Write Image Iterators can similarly be considered as a FIFO that is written to by a process. The process writes
5 pixels in a specific order to write out the entire image.

As illustrated in Fig.11, typically a process 140 will have its input tied to a Read Iterator 141, and output tied to a corresponding Write Iterator 142.

A variety of Image Iterators exist to cope with the
10 most common addressing requirements of image processing algorithms. In most cases there is a corresponding Write Iterator for each Read Iterator. The different Iterators are listed in the following table:

Read Iterators	Write Iterators
Sequential Read	Sequential Write
Box Read	-
Vertical Strip Read	Vertical Strip Write

In general, more Read Iterators are required than Write
15 Iterators. In the ACP there are 5 Read Iterators and only 3 Write Iterators.

Although an Iterator is perceived to be an unlimited FIFO, in practice there is a small FIFO connected to two or more cache lines. The small FIFO is required to allow for
20 the fact that more than one Iterator is likely to be in use at one time, and only one access can be made to the cache in a single cycle.

All FIFOs* belonging to Image Iterators can preferably be accessed by software as memory mapped I/O. General
25 software algorithms that may not be appropriate to be microcoded can therefore take advantage of the image access mechanisms.

Table Access

It can often be necessary to lookup values in a table.
30 The linear table address generator looks up the value next cycle (optional multiply by 2 for 16 bit entries) and puts results (8 or 16 bits) into an output FIFO. For 16 bits

the order is always same (lo/hi or hi/lo). The value is written to FIFO in cycle N, first 8 bits available from FIFO at start of N+2 (i.e. skips one cycle).

CCD Image Access

5 Random Access to pixels

There is no special address generator for specifying fast access to CCD images in DRAM. If a process requires random access it must directly address DRAM and decode image pixels itself.

10 Sequential Read and Sequential Write Iterators

The simplest Image Iterator is the Sequential Read Iterator. It presents the pixels from a channel one line at a time from top to bottom, and within a line, pixels are presented left to right. The padding bytes are not presented
15 to the client. It is most useful for algorithms that must perform some process on each pixel from an image but don't care about the order of the pixels being processed.

Complementing the Sequential Read Iterator is the Sequential Write Iterator. Clients write pixels to an Output
20 FIFO. A Sequential Write Iterator subsequently writes out a valid image using appropriate caching and appropriate padding bytes. Sequential Iterators requires access to 2 cache lines. When reading, while 32 pixels are presented from one cache line, the other cache line can be loaded from
25 memory.

A process that performs an operation on each pixel of an image independently would typically use a Sequential Read Iterator to obtain pixels, and a Sequential Write Iterator to write the new pixel values to their corresponding
30 locations within the destination image. It is valid to have the source image and destination image to be the same, since a given input pixel is not read more than once.

Internal Format Image Access

Further, as on a single cycle 4 bytes can be
35 transferred from an Iterator's cache into the FIFO, this allows up to 4 Iterators to do the same thing if cache

accesses are staggered. The net effect is that 4 Iterator
FIFOs can be accessed every clock cycle without the caches
having to support multiple accesses per cycle. 4 Iterators
may be 3 Read Iterators and one Write Iterator. For example,
5 as shown in Fig. 12, in a single cycle it is possible to
read 3 pixels, 1 from each of 3 Read Iterators 145-147,
perform some processing on them 148, and take the single
pixel output (derived from a previously read 3 pixels) and
transfer it to a Write Iterator 149. The average processing
10 time for a single pixel in output would thus be 1 cycle.

A variety of Image Iterators exist to cope with the
most common addressing requirements of image processing
algorithms. They are:

- Sequential Read (previously discussed)
- 15 Sequential Write (previously discussed)
- Box Read
- Vertical Strip Read
- Vertical-Strip Write

Box Read Iterator

20 The Box Read Iterator is used to present pixels in an
order most useful for performing general-purpose filters,
convolves and the like. The Iterator presents pixel values
in a square box around the sequentially read pixels. The box
is limited to being 3, 5, or 7 pixels wide. The client has
25 the choice of duplicating edge pixels, or having non-image
pixels to be a constant value. The client also has the
option of starting the center pixel of a group.

A special purpose register `IteratorSpecific1` has the
following bit usage:

Bits	Name	Usage
0	Duplicate Edge Pixels	1 = duplicate edge pixels for box region outside image 0 = return Outside Image Pixel for box region outside image
1-	Outside Image	Constant pixel value to return

8	Pixel	for pixels outside the actual image area if DuplicateEdgePixels = 0.
9-11	Reserved	-

In addition, a special purpose register AGSMSpecific1 is used to determine a sub-sampling in terms of which input pixels will be used as the center of the box. The usual value is 1, which means that each pixel is used as the center of the box. The value "2" would be useful in scaling an image down by 4:1 as in the case of building an image pyramid.

In Fig.13 there is shown a first example of the box read iterator output with Fig.14 showing a second example. In Fig.13, a box region, e.g. 150, is output for a current input pixel 151 with Fig.13 illustrating the 3x3 pixel output case. A first series of pixels 152 illustrates the box read iterator output for the current pixel 151 when duplication of edge pixels is set. A second series of output pixels 153 illustrates the case when duplication of edge pixels is not set. In this case, a pre-set constant "outside image" pixel value is output. Fig.14 illustrates a similar case for the current pixel 156 having a 3x3 output grid 155.

As illustrated in Fig.15, a process that uses the Box Read Iterator 160 for input would most likely use the Sequential Write Iterator 161 for output since they are in sync. A good example is the convolver function 162, where N input pixels are read to calculate 1 output pixel.

The Box Read Iterator will require a maximum of 14 (2 x 7) cache lines and a small (5 bytes) FIFO. While pixels are presented from one set of cache lines, the other cache lines can be loaded from memory.

30 Vertical-Strip Read and Write Iterators

In some instances it is necessary to write an image in

output pixel order, with no knowledge about the direction of coherence in input pixels in relation to output pixels.

Examples of this are rotation and warping. If it is necessary to rotate an image 90 degrees, and process the output pixels horizontally, a complete loss of cache coherence may result. On the other hand, if it is necessary to process the output image one cache line's width of pixels at a time and then advance to the next line (rather than advance to the next cache-line's worth of pixels on the same line), there will be a gain in cache coherence for some input image pixels.

It can also be the case that there is a known 'block' coherence in the input pixels (such as color coherence), in which case the read governs the processing order, and the write, to be synchronized, must follow the same pixel order.

With the vertical strip Iterators, the order of pixels presented as input (Vertical-Strip Read), or expected for output (Vertical-Strip Write) is the same and is depicted in Fig. 16. The order is pixels 0 to 31 (165) from line 0 (166), then pixels 0 to 31 of line 1 (167) etc., for all lines of the image, thereby making up first strip 169, then pixels 32 to 63 of line 0, pixels 32 to 63 of line 1 etc., making up second strip 170. In the final vertical strip there may not be exactly 32 pixels wide. In this case only the actual pixels in the image are presented or expected as input.

Referring to Fig.17, a process that requires only a Vertical-Strip Write Iterator will typically have a way of mapping input pixel coordinates given an output pixel coordinate. It would access the input image pixels according to this mapping, and coherence is determined by having sufficient cache lines on the 'random-access' reader for the input image.

It is not meaningful to pair this Write Iterator with a Sequential Read Iterator or a Box read Iterator, but a Vertical-Strip Write Iterator does give significant

improvements in performance in certain situations.

Clients read pixels from the FIFO owned by the Vertical-Strip Read Iterator that reads images cached appropriately. Clients write pixels to the FIFO owned by the Vertical-Strip Write Iterator that subsequent writes out a valid image using appropriate caching and appropriate padding bytes. Each Iterators requires 2 cache lines, and a small (5 byte) FIFO.

Table I/O Units

10 It is often necessary to lookup values in a table (which may also be an image).

Table I/O addressing modes require the client to pass an index in to an Output FIFO 76. The address generator processes the index, looks up the data appropriately, and returns the values in an Input FIFO 75 for subsequent processing by the VLIW client. 1D, 2D and 3D tables are supported, with particular modes targeted at interpolation. The 4 bit Address Mode Register is used to determine the I/O type:

Bit #	Address Mode
3	1 = This addressing mode is Table I/O
2 to 0	000 = 1D Direct lookup 001 = 1D Interpolate (linear) 010 = DRAM FIFO 011 = Reserved 100 = 2D Interpolate (bi-linear) 101 = Reserved 110 = 3D Interpolate (tri-linear) 111 = Image Pyramid lookup

The access specific registers are:

Register Name	LocalName	Description
AccessSpecific ₁	DataSize	0 = 8 bit data

		1 = 16 bit data
AccessSpecific ₂	Xmask	Point/Mask for X index
AccessSpecific ₃	Ymask	Point/Mask for Y index
AccessSpecific ₄	Zmask	Point/Mask for Z index

1 Dimensional Tables

Direct Lookup

A direct lookup is a simple index into a 1 dimensional
5 lookup table. The index passed in by the client via the
Input FIFO 75 is shifted to the appropriate location using a
Barrel Shifter, ANDed with a mask, and then ORed with the
Base Address to give the final address. The 8 or 16 bit
data value at the address is placed into the Output FIFO.
10 Address generation takes 1 cycle, and transferring the
requested data from the cache to the Input FIFO also takes 1
cycle (assuming a cache hit).

Interpolate table

This is the same as a linear table except that 2 values
15 are returned for a given address: The value returned are
Table[X], and Table[X+1] if either indexes are out of bounds
the DuplicateEdge register determines whether the edge pixel
is duplicated or a constant value is returned. Address
generation takes 1 cycle, and transferring the requested
20 data from the cache to the Input FIFO takes 2 cycles
(assuming a cache hit).

DRAM FIFO

A special case of a 1D table is a DRAM FIFO. It is
often necessary to have a simulated FIFO of a given length
25 using DRAM and associated caches. With a DRAM FIFO, clients
do not index explicitly into the table, but read and write
to the table as if it were a large FIFO. Two counters keep
track of input and output positions in the simulated FIFO,
and cache to DRAM as needed. When values are taken from the
30 Input FIFO by the client, the next values are placed into
the FIFO from the cache. When values are placed into the
Output FIFO 76 by the client, they are placed into the cache

at the next position.

2 Dimensional Tables

Direct Lookup

5 All cases of 2D lookups are needed for bi-linear interpolation.

Bi-Linear lookup

This kind of lookup is necessary for bi-linear interpolation. Given an X and Y coordinate in a table, 4 values are returned after lookup. The four values (in order) are:

Table[X, Y]
Table[X+1, Y]
Table[X, Y+1]
Table[X+1, Y+1]

15 The order specified allows for the best cache coherence.

3 Dimensional Lookup

Direct Lookup

All cases of 3D lookups are needed for tri-linear interpolation.

Tri-linear lookup

This kind of lookup is necessary for tri-linear interpolation. Given an X, Y, and Z coordinate, 8 values are returned in order from the lookup table:

25 Table[X, Y, Z]
Table[X+1, Y, Z]
Table[X, Y+1, Z]
Table[X+1, Y+1, Z]
Table[X, Y, Z+1]
Table[X+1, Y, Z+1]
30 Table[X, Y+1, Z+1]
Table[X+1, Y+1, Z+1]

The 3 values passed in by the client are barrel shifted, ORed together with the base address, and looked up.
35 The 8 sets of 1 byte values are returned via the Output FIFO.

Image Pyramid Access

During brushing, tiling, and warping it is often necessary to compute the average color of a particular area in an image. Rather than calculate the value for each area
5 given, these functions make use of an image pyramid as previously illustrated in Fig. 7. An image pyramid is effectively a multi-resolution pixel-map. The original image is a 1:1 representation. Low-pass filtering and sub-sampling by 2:1 in each dimension produces an image $\frac{1}{4}$ the original
10 size. This process continues until the entire image is represented by a single pixel.

To access an image pyramid a list of image level addresses is required. These are 12 x 32 bit registers, each stores the address of a given level in the pyramid in the
15 RDRAM memory. The width and height of the original image (level 0) is also required.

The client specifies a pixel address in terms of 3 components: x, y, and level. On subsequent cycles, 8 pixel units are returned in a specific order via a FIFO:
20 The pixel at (INTEGER[scaled x], INTEGER[scaled y], z)
The pixel at (INTEGER[scaled x]+1, INTEGER[scaled y], z)
The pixel at (INTEGER[scaled x], INTEGER[scaled y]+1, z)
The pixel at (INTEGER[scaled x]+1, INTEGER[scaled y]+1, z)
The pixel at (INTEGER[scaled x], INTEGER[scaled y], z+1)
25 The pixel at (INTEGER[scaled x]+1, INTEGER[scaled y], z+1)
The pixel at (INTEGER[scaled x], INTEGER[scaled y]+1, z+1)
The pixel at (INTEGER[scaled x]+1, INTEGER[scaled y]+1, z+1)

These 8 pixels are returned as 4 x 16 bit entries, with X & X+1 entries combined lo/hi. To access an image pyramid,
30 a list of image level addresses is required. These are 12 x 32 bit entries at a particular address given by the register ImageStart. Each entry is the address of a given level in the pyramid. The width and height of the original image (level 0) is also required, given by ImageHeight and
35 ImageWidth registers respectively. Since this access method does not write to image pyramids, the Write cache group and

write cache bus specifications are used to lookup the Image Pyramid Address Table. Only 2 cache lines should be allocated to the image level address table cache group.

5 The offset from the start of an image to a given (x, y) coordinate is given by: $\text{RowBytes} * Y + X$.
For a different level of the pyramid, a simple barrel shift right of the RowBytes value by the level number gives the RowBytes value for that level. This value needs to be multiplied by a scaled Y (also barrel shifted) and the
10 result added to a barrel shifted X value. For example, if the scaled (X, Y) coordinate was (10.4, 12.7) 4 pixels would be returned in the order (10, 12), (11, 12), (10, 13) and (11, 13). When pixels are exactly aligned (no fractional component), the "+1" pixels are duplicated (to save a read
15 from DRAM). When a coordinate is outside the valid range, clients have the choice of edge pixel duplication or returning of a constant color value via the DuplicateEdgePixels and ConstantPixel registers (only the low 8 bits are used).

20 Generation of Coordinates using VLIW Vector Processor

Some functions that are linked to Write Iterators require the X and Y coordinates of the current pixel being processed in part of the processing pipeline. Particular processing may need to take place at the end of each row, or
25 column being processed.

Each function requiring coordinates will have a different pixel calculation time, and as such will have slightly different timing for coordinate generation. However, The essence and ALU requirements will be the same
30 in each instance, however.

Generate Sequential [X, Y]

When a process is processing pixels in sequential order according to the Sequential Read Iterator (or generating pixels and writing them out to a Sequential Write Iterator),
35 the process as shown in Fig. 28 can be used to generate X, Y coordinates. One form of implementation is as shown in Fig.

28. The coordinate generator counts up to ImageWidth in the X ordinate, and once per ImageWidth pixels increments the Y ordinate. The following constants of Fig. 29 are set by software:

Constant	Value
K ₁	ImageWidth
K ₂	ImageHeight (optional)

5

The following registers are used to hold temporary variables:

Variable	Value
Latch ₁	X (starts at 0 each line)
Latch ₂	Y (starts at 0)

Generate Vertical Strip [X, Y]

10 The vertical strip generation process is as shown in Fig. 30. The coordinate generator simply counts up to ImageWidth in the X ordinate, and once per ImageWidth pixels increments the Y ordinate. An actual implementation is as illustrated in Fig. 31, where the following constants are
15 set by software:

Constant	Value
K ₁	32
K ₂	ImageWidth
K ₃	ImageHeight

The following registers are used to hold temporary variables:

Variable	Value
Latch ₁	StartX (starts at 0, and is incremented by 32 once per vertical strip)
Latch ₂	X
Latch ₃	EndX (starts at 32 and is incremented by 32 to a maximum of ImageWidth) once per vertical strip)

Latch ₄	Y
--------------------	---

Display Controller 88

Principles of Operation

When the "Take" button on an Artcam is half depressed,
5 the TFT will display the current image from the image sensor
(converted via a simple VLIW process). Once the Take button
is fully depressed, the Taken Image is displayed.

When the user presses the Print button and image
processing begins, the TFT can be turned off to avoid
10 excessive power drain. Once the image has been printed the
TFT is turned on again.

Structural Overview

The Display Controller 88 of Fig. 3 is used in those
Artcam models that incorporate a flat panel display. An
15 example display is a TFT LCD of resolution 240 x 160 pixels.
The structure of the Display Controller is as illustrated in
Fig. 32.

The Display Controller State Machine contains registers
that control the timing of the Sync Generation, where the
20 display image is to be taken from (in DRAM via the Cache
Interface), and whether the TFT should be active or not (via
a TFT Enable signal) at the moment. The CPU can write to
those registers via the low speed bus.

Displaying a 240 x 160 pixel image on an RGB TFT
25 requires 3 components per pixel. The image taken from DRAM
is displayed via 3 DACs, one for each of the R, G, and B
output signals.

At an image refresh rate of 30 frames per second (60
fields per second) the Display Controller requires data
30 transfer rates of:

$$240 \times 160 \times 3 \times 30 = 3.5\text{MB per second}$$

This data rate is low compared to the rest of the
system. However it is high enough to cause VLIW programs to
slow down during the intensive image processing. The general
35 principles of TFT operation should reflect this.

CPU Core (CPU) 72

The CPU core 72 can be any processor core with sufficient processing power to perform the required core calculations and control functions fast enough to meet
5 consumer expectations. Examples of suitable cores are:

MIPS R4000 core from LSI Logic

StrongARM core

The Artcam is deliberately designed so that the core processor 72 can be changed at any stage while maintaining
10 complete compatibility. To use a different core, the Vark interpreter and camera control programs must be re-compiled for the new processor instruction set. This is a straightforward task if the Vark interpreter is written in a high level language (preferably C++) with no assembler.

15 The Vark language preferably makes no assumptions about the CPU, and is completely portable. Therefore any Artcards will work with any CPU cores which meet the performance specifications. As a result of this device independence, future Artcam models can take advantage of new processor
20 cores as they are developed. Also, different ACP chip designs may be fabricated by different manufacturers, without the need to license or port the CPU core.

Authentication chip serial interfaces 64

These are 2 standard low-speed serial ports, which are
25 connected to the internal chip low speed bus. The reason for having 2 ports is to connect to both the on-camera Authentication chip, and to the print-roll Authentication chip using separate lines. Only using 1 line may make it possible for a clone print-roll manufacturer to design a
30 chip which, instead of generating an authentication code, tricks the camera into using the code generated by the authentication chip in the camera.

Parallel Interface 65

The parallel interface connects the ACP to individual static
35 electrical signals. The following is a table of connections to the parallel interface:

Connection	Direction	Pins
Paper transport stepper motor	Output	4
Artcard stepper motor	Output	4
Zoom stepper motor	Output	4
Guillotine solenoid	Output	1
Flash trigger	Output	1
Status LCD segment drivers	Output	7
Status LCD common drivers	Output	4
Artcard illumination LED	Output	1
Artcard status LED (red/green)	Input	2
Artcard sensor	Input	1
Paper pull sensor	Input	1
Orientation sensor	Input	2
Buttons	Input	4
Total		36

Print Head Interface 62

Once an image has been processed, it can be printed. The Print Head Interface connects the ACP to the Print Head, providing both data and appropriate signals to the external Print Head.

Image Sensor Interface (ISI) 83

The Image Sensor Interface (ISI) 83 of Fig. 3 takes data from the CCD and makes it available for storage in DRAM. The CCD can be a 3:2 aspect ratio image sensor, typically 750 x 500, yielding 375K (8 bits per pixel). Fig. 4 illustrates the configuration of a single pixel.

The ISI 83 can include a state machine that sends control information to the CCD 2 (Fig.2), including frame sync pulses and pixel clock pulses in order to read the image. Pixels are read from the CCD via a sub-ranging semi-flash DAC, and placed into the VLIW Input FIFO. The VLIW is then able to process and/or store the pixels, which are then

available for processing and/or storage.

The ISI 83 is used in conjunction with a VLIW microcode program that stores the CCD image in DRAM. Processing occurs in 2 steps:

5 1. A small VLIW program reads the pixels from the FIFO 192 and writes them to the DRAM via a Sequential Write Iterator.

2. The CCD image in DRAM is rotated 90, 180 or 270 degrees according to the orientation of the camera when the photo was taken.

10 If the rotation is 0 degrees, then step 1 merely writes the CCD image out to the final CCD image location and step 2 is not performed. If the rotation is non-0 degrees, the image is written out to a temporary area (for example into the print image memory area), and then rotated during step 2
15 into the final CCD image location. Step 1 is very simple microcode, taking data from the VLIW Input FIFO 192 and writing it to a Sequential Write Iterator. Step 2's rotation is accomplished by using the accelerated Vark Affine Transform function. The processing is performed in 2 steps
20 in order to reduce design complexity and to re-use the Vark affine transform rotate logic already required for images. This is acceptable since both steps are completed in less than 0.03 seconds, a time imperceptible to the operator of the Artcam. Even so, the read process is CCD speed bound,
25 taking 0.02 seconds to read the full frame. The time taken to rotate the image can be 2 cycles per output pixel, which is 750,000 cycles, or 0.008 seconds. The total time for both stages is therefore 0.028 seconds.

30 The orientation will be important for converting between the CCD image and the internal format image, since the relative positioning of R, G, and B pixels changes with orientation. The processed image may also have to be rotated during the Print process in order to be in the correct orientation for printing.

35 On the optional 3D model of the Artcam there are 2 CCDs, with their inputs multiplexed to a single ISI

(different microcode, but same ACP). If the CCD has a frame store both frames can be taken simultaneously, and then transferred to memory one at a time. If the CCD has a line store, the frames can be transferred one line at a time in a multiplexed fashion.

Artcard Interface (AI) 87

The Artcard Interface (AI) 87 is responsible for taking an Artcard image from the Artcard Reader 34 , and decoding it into the original data (usually a Vark script).

Specifically, the AI 87 accepts signals from the Artcard scanner linear CCD 34, detects the bit pattern printed on the card, and converts the bit pattern into the original data, correcting read errors.

With no Artcard 9 inserted, the image printed from an Artcam is simply the sensed Photo Image cleaned up by any standard image processing routines. The Artcard 9 is the means by which users are able to modify a photo before printing it out. By the simple task of inserting a specific Artcard 9 into an Artcam, a user is able to define complex image processing to be performed on the Photo Image. With no Artcard inserted the Photo Image is processed in a standard way to create the Print Image. When a single Artcard 9 is inserted into the Artcam, that Artcard's effect is applied to the Photo Image to generate the Print Image. When the Artcard 9 is removed (ejected), the printed image reverts to the Photo Image processed in a standard way. When the user presses the button to eject an Artcard, an event is placed in the event queue maintained by the operating system running on the Artcam Central Processor 31. When the event is processed (for example after the current Print has occurred), the following things occur:

If the current Artcard is valid, then the Print Image is marked as invalid and a 'Process Standard' event is placed in the event queue. When the event is eventually processed it will perform the standard image processing operations on the Photo Image to produce the Print Image.

The motor is started to eject the Artcard and a time-specific 'Stop-Motor' Event is added to the event queue.
Inserting an Artcard

When a user inserts an Artcard 9, the Artcard Sensor 49
5 detects it notifying the ACP72. This results in the software inserting an 'Artcard Inserted' event into the event queue. When the event is processed several things occur:

The current Artcard is marked as invalid (as opposed to
10 'none').

The Print Image is marked as invalid.

The Artcard motor 37 is started up to load the Artcard

The Artcard Interface 87 is instructed to read the
Artcard

15 The Artcard Interface 87 accepts signals from the Artcard scanner linear CCD 34, detects the bit pattern printed on the card, and corrects errors in the detected bit pattern, producing a valid Artcard data block in DRAM.

Reading Data from the Artcard CCD - General Considerations

20 As illustrated in Fig. 33, the Data Card reading process has 4 phases operated while the pixel data is read from the card. The phases are as follows:

Phase 1. Detect data area on Artcard

Phase 2. Detect bit pattern from Artcard based
25 on CCD pixels, and write as bytes.

Phase 3. Descramble and XOR the byte-pattern

Phase 4. Decode data (Reed-Solomon decode)

As illustrated in Fig. 34, the Artcard 9 must be
sampled at least at double the printed resolution to satisfy
30 Nyquist's Theorem. In practice it is better to sample at a higher rate than this. Preferably, the pixels are sampled 230 at 3 times the resolution of a printed dot in each dimension, requiring 9 pixels to define a single dot. Thus if the resolution of the Artcard 9 is 1600 dpi, and the
35 resolution of the sensor 34 is 4800 dpi, then using a 50mm CCD image sensor results in 9450 pixels per column.

Therefore if we require 2MB of dot data (at 9 pixels per dot) then this requires $2\text{MB} \times 8 \times 9 / 9450 = 15,978$ columns = approximately 16,000 columns. Of course if a dot is not exactly aligned with the sampling CCD the worst and most likely case is that a dot will be sensed over a 16 pixel area (4x4) 231.

An Artcard 9 may be slightly warped due to heat damage, slightly rotated (up to, say 1 degree) due to differences in insertion into an Artcard reader, and can have slight differences in true data rate due to fluctuations in the speed of the reader motor 37. These changes will cause columns of data from the card not to be read as corresponding columns of pixel data. As illustrated in Fig. 35, a 1 degree rotation in the Artcard 9 can cause the pixels from a column on the card to be read as pixels across 166 columns:

Finally, the Artcard 9 should be read in a reasonable amount of time with respect to the human operator. The data on the Artcard covers most of the Artcard surface, so timing concerns can be limited to the Artcard data itself. A reading time of 1.5 seconds is adequate for Artcard reading.

The Artcard should be loaded in 1.5 seconds. Therefore all 16,000 columns of pixel data must be read from the CCD 34 in 1.5 second, i.e. 10,667 columns per second. Therefore the time available to read one column is $1/10667$ seconds, or 93,747ns. Pixel data can be written to the DRAM one column at a time, completely independently from any processes that are reading the pixel data.

The time to write one column of data (9450/2 bytes since the reading can be 4 bits per pixel giving 2×4 bit pixels per byte) to DRAM is reduced by using 8 cache lines. If 4 lines were written out at one time, the 4 banks can be written to independently and thus overlap latency reduced. Thus the 4725 bytes can be written in 11,840ns ($4725/128 \times 320\text{ns}$). Thus the time taken to write a given column's data to DRAM uses just under 13% of the available bandwidth.

Decoding an Artcard

A simple look at the data sizes shows the impossibility of fitting the process into the 8MB of memory 33 if the entire Artcard pixel data (140 MB if each bit is read as a 3x3 array) as read by the linear CCD 34 is kept. For this reason, the reading of the linear CCD, decoding of the bitmap, and the un-bitmap process should take place in real-time (while the Artcard 9 is traveling past the linear CCD 34), and these processes must effectively work without having entire data stores available.

When an Artcard 9 is inserted, the old stored Print Image and any expanded Photo Image becomes invalid. The new Artcard 9 can contain directions for creating a new image based on the currently captured Photo Image. The old Print Image is invalid, and the area holding expanded Photo Image data and image pyramid is invalid, leaving more than 5MB that can be used as scratch memory during the read process. Strictly speaking, the 1MB area where the Artcard raw data is to be written can also be used as scratch data during the Artcard read process as long as by the time the final Reed-Solomon decode is to occur, that 1MB area is free again. The reading process described here does not make use of the extra 1MB area (except as a final destination for the data).

It should also be noted that the unscrambling process requires two sets of 2MB areas of memory since unscrambling cannot occur in place. Fortunately the 5MB scratch area contains enough space for this process.

Turning now to Fig. 36, there is shown a flowchart of the steps necessary to decode the Artcard data. These steps include reading in the Artcard data, decoding the read data to produce corresponding encoded XORed scrambled bitmap data. Next a checkerboard XOR is applied to the data to produces encoded scrambled data. This data is then unscrambled to produce data before this data is subjected to Reed-Solomon decoding to produce the original raw data. Alternatively, unscrambling and XOR process

can take place together, not requiring a separate pass of the data. Each of the above steps is discussed in further detail hereinafter. As noted previously with reference to Fig. 36, the Artcard Interface, therefore, has 4 phases, the first 2 of which are time-critical, and must take place while pixel data is being read from the CCD:

- Phase 1. Detect data area on Artcard
- Phase 2. Detect bit pattern from Artcard based on CCD pixels, and write as bytes.
- 10 Phase 3. Descramble and XOR the byte-pattern
- Phase 4. Decode data (Reed-Solomon decode)

The four phases are described in more detail as follows:

Phase 1. As the Artcard 9 moves past the CCD 34 the AI must detect the start of the data area by robustly detecting special targets on the Artcard to the left of the data area. If these cannot be detected, the card is marked as invalid. The detection must occur in real-time, while the Artcard 9 is moving past the CCD 34.

20 If necessary, rotation invariance can be provided. In this case, the targets are repeated on the right side of the Artcard, but relative to the bottom right corner instead of the top corner. In this way the targets end up in the correct orientation if the card is inserted the "wrong" way.

25 Phase 3 below can be altered to detect the orientation of the data, and account for the potential rotation.

Phase 2. Once the data area has been determined, the main read process begins, placing pixel data from the CCD into an 'Artcard data window', detecting bits from this window, assembling the detected bits into bytes, and constructing a byte-image in DRAM. This must all be done while the Artcard is moving past the CCD.

Phase 3. Once all the pixels have been read from the Artcard data area, the Artcard motor 37 can be stopped, and the byte image descrambled and XORed. Although not requiring real-time performance, the process should be fast enough not

to annoy the human operator. The process must take 2 MB of scrambled bit-image and write the unscrambled/XORed bit-image to a separate 2MB image.

5 Phase 4. The final phase in the Artcard read process is the Reed-Solomon decoding process, where the 2MB bit-image is decoded into a 1MB valid Artcard data area. Again, while not requiring real-time performance it is still necessary to decode quickly with regard to the human operator. If the decode process is valid, the card is marked as valid. If the
10 decode failed, any duplicates of data in the bit-image are attempted to be decoded, a process that is repeated until success or until there are no more duplicate images of the data in the bit image.

The four phase process described requires 4.5 MB of
15 DRAM. 2MB is reserved for Phase 2 output, and 0.5MB is reserved for scratch data during phases 1 and 2. The remaining 2MB of space can hold over 440 columns at 4725 bytes per column. In practice, the pixel data being read is a few columns ahead of the phase 1 algorithm, and in the worst
20 case, about 180 columns behind phase 2, comfortably inside the 440 column limit.

A description of the actual operation of each phase will now be provided in greater detail.

Phase 1 - Detect data area on Artcard

25 This phase is concerned with robustly detecting the left-hand side of the data area on the Artcard 9. Accurate detection of the data area is achieved by accurate detection of special targets printed on the left side of the card. These targets are especially designed to be easy to detect
30 even if rotated up to 1 degree.

Turning to Fig.37, there is shown an enlargement of the left hand side of an Artcard 9. The side of the card is divided into 16 bands, 239 with a target eg. 241 located at the center of each band. The bands are logical in that there
35 is no line drawn to separate bands. Turning to Fig.38, there is shown a single target 241. The target 241, is a printed

black square containing a single white dot. The idea is to detect firstly as many targets 241 as possible, and then to join at least 8 of the detected white-dot locations into a single logical straight line. If this can be done, the
5 start of the data area 243 is a fixed distance from this logical line. If it cannot be done, then the card is rejected as invalid.

As shown in Fig. 37, the height of the card 9 is 3150 dots. A target (Target0) 241 is placed a fixed distance of
10 24 dots away from the top left corner 244 of the data area so that it falls well within the first of 16 equal sized regions 239 of 192 dots (576 pixels) with no target in the final pixel region of the card. The target 241 must be big enough to be easy to detect, yet be small enough not to go
15 outside the height of the region if the card is rotated 1 degree. A suitable size for the target is a 31 x 31 dot (93 x 93 sensed pixels) black square 241 with the white dot 242.

At the worst rotation of 1 degree, a 1 column shift occurs every 57 pixels. Therefore in a 590 pixel sized band,
20 we cannot place any part of our symbol in the top or bottom 12 pixels or so of the band or they could be detected in the wrong band at CCD read time if the card is worst case rotated.

Therefore, if the black part of the rectangle is 57
25 pixels high (19 dots) we can be sure that at least 9.5 black pixels will be read in the same column by the CCD (worst case is half the pixels are in one column and half in the next). To be sure of reading at least 10 black dots in the same column, we must have a height of 20 dots. To give room
30 for erroneous detection on the edge of the start of the black dots, we increase the number of dots to 31, giving us 15 on either side of the white dot at the target's local coordinate (15, 15). 31 dots is 91 pixels, which at most suffers a 3 pixel shift in column, easily within the 576
35 pixel band.

Thus each target is a block of 31 x 31 dots (93 x 93

pixels) each with the composition:

15 columns of 31 black dots each (45 pixel width
columns of 93 pixels).

1 column of 15 black dots (45 pixels) followed by 1
5 white dot (3 pixels) and then a further 15 black dots (45
pixels)

15 columns of 31 black dots each (45 pixel width
columns of 93 pixels)

Detect targets

10 Targets are detected by reading columns of pixels, one
column at a time rather than by detecting dots. It is
necessary to look within a given band for a number of
columns consisting of large numbers of contiguous black
pixels to build up the left side of a target. Next, it is
15 expected to see a white region in the center of further
black columns, and finally the black columns to the left of
the target center.

Eight cache lines are required for good cache
performance on the reading of the pixels. Each logical read
20 fills 4 cache lines via 4 sub-reads while the other 4 cache-
lines are being used. This effectively uses up 13% of the
available DRAM bandwidth.

As illustrated in Fig.39, the detection mechanism FIFO
for detecting the targets uses a filter 245, run-length
25 encoder 246, and a FIFO 247 that requires special wiring of
the top 3 elements (S1, S2, and S3) for random access.

The columns of input pixels are processed one at a time
until either all the targets are found, or until a specified
number of columns have been processed. To process a column,
30 the pixels are read from DRAM, passed through a filter 245
to detect a 0 or 1, and then run length encoded 246. The bit
value and the number of contiguous bits of the same value
are placed in FIFO 247. Each entry of the FIFO 249 is in 8
bits, 7 bits 250 to hold the run-length, and 1 bit 249 to
35 hold the value of the bit detected.

The run-length encoder 246 only encodes contiguous

pixels within a 576 pixel (192 dot) region.

The top 3 elements in the FIFO 247 can be accessed 252 in any random order. The run lengths (in pixels) of these entries are filtered into 3 values: *short*, *medium*, and *long* in accordance with the following table:

Short	Used to detect white dot.	RunLength < 16
Medium	Used to detect runs of black above or below the white dot in the center of the target.	16<= RunLength < 48
Long	Used to detect run lengths of black to the left and right of the center dot in the target.	RunLength >= 48

Looking at the top three entries in the FIFO 247 there are 3 specific cases of interest:

10

Case 1	S1 = white long S2 = black long S3 = white medium/long	We have detected a black column of the target to the left of or to the right of the white center dot.
Case 2	S1 = white long S2 = black medium S3 = white short Previous 8 columns were Case 1	If we've been processing a series of columns of Case 1s, then we have probably detected the white dot in this column. We know that the next entry will be black (or it would have been included in the white S3 entry), but the number of black pixels is in question. Need to verify by checking after the next FIFO advance

		(see Case 3).
Case 3	Prev = Case 2 S3 = black med	We have detected part of the white dot. We expect around 3 of these, and then some more columns of Case 1.

Preferably, the following information per region band is kept:

TargetDetected	1 bit
BlackDetectCount	4 bits
WhiteDetectCount	3 bits
PrevColumnStartPixel	15 bits
TargetColumn ordinate	16 bits (15:1)
TargetRow ordinate	16 bits (15:1)
TOTAL	7 bytes (rounded to 8 bytes for easy addressing)

5

Given a total of 7 bytes. It makes address generation easier if the total is assumed to be 8 bytes. Thus 16 entries requires $16 * 8 = 128$ bytes, which fits in 4 cache lines. The address range should be inside the scratch 0.5MB DRAM area since other phases make use of the remaining 4MB data area.

10

When beginning to process a given pixel column, the register value S2StartPixel 254 is reset to 0. As entries in the FIFO advance from S2 to S1, they are also added 255 to the existing S2StartPixel value, giving the exact pixel position of the run currently defined in S2. Looking at each of the 3 cases of interest in the FIFO, S2StartPixel can be used to determine the start of the black area of a target (Cases 1 and 2), and also the start of the white dot in the

15

center of the target (Case 3). An algorithm for processing columns can be as follows:

1	TargetDetected[0-15] := 0 BlackDetectCount[0-15] := 0 WhiteDetectCount[0-15] := 0 TargetRow[0-15] := 0 TargetColumn[0-15] := 0 PrevColStartPixel[0-15] := 0 CurrentColumn := 0
2	Do ProcessColumn
3	CurrentColumn++
4	If (CurrentColumn <= LastValidColumn) Goto 2

- 5 The steps involved in the processing a column (Process Column) are as follows:

1	S2StartPixel := 0 FIFO := 0 BlackDetectCount := 0 WhiteDetectCount := 0 ThisColumnDetected := FALSE PrevCaseWasCase2 := FALSE
2	If (! TargetDetected[Target]) & (! ColumnDetected[Target]) * ProcessCases EndIf
3	PrevCaseWasCase2 := Case=2
4	Advance FIFO

- 10 The processing for each of the 3 (Process Cases) cases is as follows:

Case 1:

BlackDetectCount[target] < 8	$\Delta := \text{ABS}(\text{S2StartPixel} - \text{PrevColStartPixel}[\text{Target}])$
------------------------------	---

OR <code>WhiteDetectCount[Target] = 0</code>	<code>If (0<=Δ< 2)</code> <code>BlackDetectCount[Target]++ (max value =8)</code> <code>Else</code> <code>BlackDetectCount[Target] := 1</code> <code>WhiteDetectCount[Target] := 0</code> <code>EndIf</code> <code>PrevColStartPixel[Target] := S2StartPixel</code> <code>ColumnDetected[Target] := TRUE</code> <code>BitDetected = 1</code>
<code>BlackDetectCount[target] >= 8</code> <code>WhiteDetectCount[Target] != 0</code>	<code>PrevColStartPixel[Target] := S2StartPixel</code> <code>ColumnDetected[Target] := TRUE</code> <code>BitDetected = 1</code> <code>TargetDetected[Target] := TRUE</code> <code>TargetColumn[Target] := CurrentColumn – 8 –</code> <code>(WhiteDetectCount[Target]/2)</code>

Case 2:

No special processing is recorded except for setting the 'PrevCaseWasCase2' flag for identifying Case 3 (see Step 3 of processing a column described above)

Case 3:

<pre> PrevCaseWasCase2 = TRUE BlackDetectCount[Target] >= 8 WhiteDetectCount=1 </pre>	<pre> If (WhiteDetectCount[Target] < 2) TargetRow[Target] = S2StartPixel + (S2RunLength/2) EndIf Δ := ABS(S2StartPixel - PrevColStartPixel[Target]) If (0<=Δ< 2) WhiteDetectCount[Target]++ Else WhiteDetectCount[Target] := 1 EndIf PrevColStartPixel[Target] := S2StartPixel ThisColumnDetected := TRUE </pre>
--	---

	BitDetected = 0
--	-----------------

At the end of processing a given column, a comparison is made of the current column to the maximum number of columns for target detection. If the number of columns
5 allowed has been exceeded, then it is necessary to check how many targets have been found. If fewer than 8 have been found, the card is considered invalid.

Process targets

After the targets have been detected, they should be
10 processed. All the targets may be available or merely some of them. Some targets may also have been erroneously detected.

This phase of processing is to determine a mathematical line that passes through the center of as many targets as
15 possible. The more targets that the line passes through, the more confident the target position has been found. The limit is set to be 8 targets. If a line passes through at least 8 targets, then it is taken to be the right one.

It is all right to take a brute-force but
20 straightforward approach since there is the time to do so (see below), and lowering complexity makes testing easier. It is necessary to determine the line between targets 0 and 1 (if both targets are considered valid) and then determine how many targets fall on this line. Then we determine the
25 line between targets 0 and 2, and repeat the process. Eventually we do the same for the line between targets 1 and 2, 1 and 3 etc. and finally for the line between targets 14 and 15. Assuming all the targets have been found, we need to perform $15+14+13+ \dots = 90$ sets of calculations (with each set
30 of calculations requiring 16 tests = 1440 actual calculations), and choose the line which has the maximum number of targets found along the line. The algorithm for target location can be as follows:

TargetA := 0

```
MaxFound := 0
BestLine := 0
While (TargetA < 15)
    If (TargetA is Valid)
5      TargetB:= TargetA + 1
      While (TargetB<= 15)
        If (TargetB is valid)
          CurrentLine := line between TargetA and TargetB
          TargetC := 0;
10       While (TargetC <= 15)
          If (TargetC valid AND TargetC on line AB)
            TargetsHit++
          EndIf
          If (TargetsHit > MaxFound)
15             MaxFound := TargetsHit
             BestLine := CurrentLine
          EndIf
          TargetC++
        EndWhile
20       EndIf
      TargetB ++
      EndWhile
    EndIf
    TargetA++
25 EndWhile
```

```
    If (MaxFound < 8)
      Card is Invalid
    Else
30      Store expected centroids for rows based on BestLine
    EndIf
```

As illustrated in Fig. 33, in the algorithm above, to determine a CurrentLine 260 from Target A 261 and target B, it is necessary to calculate Δ row (264) & Δ column (263)

35 between targets 261, 262, and the location of Target A. It is then possible to move from Target 0 to Target 1 etc. by

adding Δrow and Δcolumn . The found (if actually found) location of target N can be compared to the calculated expected position of Target N on the line, and if it falls within the tolerance, then Target N is determined to be on the line.

To calculate Δrow & Δcolumn :

$$\Delta\text{row} = (\text{row}_{\text{TargetA}} - \text{row}_{\text{TargetB}}) / (B-A)$$

$$\Delta\text{column} = (\text{column}_{\text{TargetA}} - \text{column}_{\text{TargetB}}) / (B-A)$$

Then we calculate the position of Target0:

10 $\text{row} = \text{row}_{\text{TargetA}} - (A * \Delta\text{row})$

$$\text{column} = \text{column}_{\text{TargetA}} - (A * \Delta\text{column})$$

And compare (row, column) against the actual $\text{row}_{\text{Target0}}$ and $\text{column}_{\text{Target0}}$. To move from one expected target to the next (e.g. from Target0 to Target1), we simply add Δrow and

15 Δcolumn to row and column respectively. To check if each target is on the line, we must calculate the expected position of Target0, and then perform one add and one comparison for each target ordinate.

At the end of comparing all 16 targets against a maximum of 90 lines, the result is the best line through the valid targets. If that line passes through at least 8 targets (i.e. $\text{MaxFound} \geq 8$), it can be said that enough targets have been found to form a line, and thus the card can be processed. If the best line passes through fewer than 25 8, then the card is considered invalid.

The resulting algorithm takes 180 divides to calculate Δrow and Δcolumn , 180 multiply/adds to calculate target0 position, and then 2880 adds/comparisons. The time we have to perform this processing is the time taken to read 36 30 columns of pixel data = 3,374,892ns. Not even accounting for the fact that an add takes less time than a divide, it is necessary to perform 3240 mathematical operations in 3,374,892ns. That gives approximately 1040ns per operation, or 104 cycles. The CPU can therefore safely perform the

entire processing of targets, reducing complexity of design.

Update centroids based on data edge border and clockmarks

Step 0: Locate the data area

5 From Target 0 (241 of Fig.37) it is a
predetermined fixed distance in rows and columns to the top
left border 244 of the data area, and then a further 1 dot
column to the vertical clock marks 276. So we use TargetA,
Δrow and Δcolumn found in the previous stage (Δrow and
10 Δcolumn refer to distances between targets) to calculate
the centroid or expected location for Target0 as described
previously.

Since the fixed pixel offset from Target0 to the data
area is related to the distance between targets (192 dots
15 between targets, and 24 dots between Target0 and the data
area 243), simply add Δrow/8 to Target0's centroid column
coordinate (aspect ratio of dots is 1:1). Thus the top co-
ordinate can be defined as:

$$(\text{column}_{\text{DotColumnTop}} = \text{column}_{\text{Target0}} + (\Delta\text{row}/8))$$

20 $(\text{row}_{\text{DotColumnTop}} = \text{row}_{\text{Target0}} + (\Delta\text{column} / 8))$

Next Δrow and Δcolumn are updated to give the number
of pixels between dots in a single column (instead of
between targets) by dividing them by the number of dots
between targets:

25 $\Delta\text{row} = \Delta\text{row}/192$

$$\Delta\text{column}^* = \Delta\text{column} / 192$$

We also set the currentColumn register (see Phase 2) to
be -1 so that after step 2, when phase 2 begins, the
currentColumn register will increment from -1 to 0.

30 Step 1: Write out the initial centroid deltas (Δ) and bit history

This simply involves writing setup information required
for Phase 2.

This can be achieved by writing 0s to all the Δrow and

Δcolumn entries for each row, and a bit history. The bit history is actually an expected bit history since it is known that to the left of the clock mark column 276 is a border column 277, and before that, a white area. The bit history therefore is 011, 010, 011, 010 etc.

Step 2: Update the centroids based on actual pixels read.

The bit history is set up in Step 1 according to the expected clock marks and data border. The actual centroids for each dot row can now be more accurately set (they were initially 0) by comparing the expected data against the actual pixel values. The centroid updating mechanism is achieved by simply performing step 3 of Phase 2.

Phase 2 - Detect bit pattern from Artcard based on pixels read, and write as bytes.

Since a dot from the Artcard 9 requires a minimum of 9 sensed pixels over 3 columns to be represented, there is little point in performing dot detection calculations every sensed pixel column. It is better to average the time required for processing over the average dot occurrence, and thus make the most of the available processing time. This allows processing of a column of dots from an Artcard 9 in the time it takes to read 3 columns of data from the Artcard. Although the most likely case is that it takes 4 columns to represent a dot, the 4th column will be the last column of one dot and the first column of a next dot. Processing should therefore be limited to only 3 columns.

As the pixels from the CCD are written to the DRAM in 13% of the time available, 83% of the time is available for processing of 1 column of dots i.e. 83% of $(93,747 \times 3) = 83\%$ of $281,241\text{ns} = 233,430\text{ns}$.

In the available time, it is necessary to detect 3150 dots, and write their bit values into the raw data area of memory. The processing therefore requires the following steps:

For each column of dots on the Artcard:

Step 0: Advance to the next dot column

Step 1: Detect the top and bottom of an Artcard dot column (check clock marks)

Step 2: Process the dot column, detecting bits and
5 storing them appropriately

Step 3: Update the centroids

Since we are processing the Artcard's logical dot columns, and these may shift over 165 pixels, the worst case is that we cannot process the first column until at least
10 165 columns have been read into DRAM. Phase 2 would therefore finish the same amount of time after the read process had terminated. The worst case time is: $165 * 93,747\text{ns} = 15,468,255\text{ns}$ or 0.015 seconds.

Step 0: Advance to the next dot column

15 In order to advance to the next column of dots we add Δrow and Δcolumn to the dotColumnTop to give us the centroid of the dot at the top of the column. The first time we do this, we are currently at the clock marks column 276 to the left of the bit image data area, and so we advance to the
20 first column of data. Since Δrow and Δcolumn refer to distance between dots within a column, to move between dot columns it is necessary to add Δrow to $\text{column}_{\text{dotColumnTop}}$ and Δcolumn to $\text{row}_{\text{dotColumnTop}}$.

To keep track of what column number is being processed,
25 the column number is recorded in a register called CurrentColumn . Every time the sensor advances to the next dot column it is necessary to increment the CurrentColumn register. The first time it is incremented, it is incremented from -1 to 0 (see Step 0 Phase 1). The
30 CurrentColumn register determines when to terminate the read process (when reaching maxColumns), and also is used to advance the DataOut Pointer to the next column of byte information once all 8 bits have been written to the byte (once every 8 dot columns). The lower 3 bits determine what
35 bit we're up to within the current byte. It will be the same

bit being written for the whole column.

Step 1: Detect the top and bottom of an Artcard dot column.

In order to process a dot column from an Artcard, it is
5 necessary to detect the top and bottom of a column. The
column should form a straight line between the top and
bottom of the column (except for local warping etc.).
Initially dotColumnTop points to the clock mark column 276.
We simply toggle the expected value, write it out into the
10 bit history, and move on to step 2, whose first task will be
to add the Δrow and $\Delta column$ values to dotColumnTop to
arrive at the first data dot of the column.

Step 2: Process an Artcard's dot column

Given the centroids of the top and bottom of a column.
15 in pixel coordinates the column should form a straight line
between them, with possible minor variances due to warping
etc.

Assuming the processing is to start at the top of a
column (at the top centroid coordinate) and move down to the
20 bottom of the column, subsequent expected dot centroids are
given as:

$$row_{next} = row + \Delta row$$

$$column_{next} = column + \Delta column$$

This gives us the address of the expected centroid for
25 the next dot of the column. However to account for *local*
warping and error we add another Δrow and $\Delta column$ based on
the last time we found the dot in a given row. In this way
we can account for small drifts that accumulate into a
maximum drift of some percentage from the straight line
30 joining the top of the column to the bottom.

We therefore keep 2 values for each row, but store them
in separate tables since the row history is used in step 3
of this phase.

- * Δrow and $\Delta column$ (2 @ 4 bits each = 1 byte)
- 35 * row history (3 bits per row, 2 rows are stored per

byte)

For each row we need to read a Δ row and Δ column to determine the change to the centroid. The read process takes 5% of the bandwidth and 2 cache lines:

5 $76 * (3150 / 32) + 2 * 3150 = 13,824 \text{ ns} = 5\% \text{ of bandwidth}$

Once the centroid has been determined, the pixels around the centroid need to be examined to detect the status of the dot and hence the value of the bit. In the worst case a dot covers a 4x4 pixel area. However, thanks to the fact
10 that we are sampling at 3 times the resolution of the dot, the number of pixels required to detect the status of the dot and hence the bit value is much less than this. We only require access to 3 columns of pixel columns at any one time.

15 In the worst case of pixel drift due to a 1% rotation, centroids will shift 1 column every 57 pixel rows, but since a dot is 3 pixels in diameter, a given column will be valid for 171 pixel rows (3*57). As a byte contains 2 pixels, the number of bytes valid in each buffered read (4 cache lines)
20 will be a worst case of 86 (out of 128 read).

Once the bit has been detected it must be written out to DRAM. We store the bits from 8 columns as a set of contiguous bytes to minimize DRAM delay. Since all the bits from a given dot column will correspond to the next bit
25 position in a data byte, we can read the old value for the byte, shift and OR in the new bit, and write the byte back.

The read / shift&OR / write process requires 2 cache lines.

We need to read and write the bit history for the given
30 row as we update it. We only require 3 bits of history per row, allowing the storage of 2 rows of history in a single byte. The read / shift&OR / write process requires 2 cache lines.

The total bandwidth required for the bit detection and
35 storage is summarised in the following table:

Read centroid Δ	5%
Read 3 columns of pixel data	19%
Read/Write detected bits into byte buffer	10%
Read/Write bit history	5%
TOTAL	39%

Detecting a dot

5 The process of detecting the value of a dot (and hence the value of a bit) given a centroid is accomplished by examining 3 pixel values and getting the result from a lookup table. The process is fairly simple and is illustrated in Fig. 41. A dot 290 has a radius of about 1.5 pixels. Therefore the pixel 291 that holds the centroid, regardless of the actual position of the centroid within that pixel, should be 100% of the dot's value. If the centroid is exactly in the center of the pixel 291, then the pixels above 292 & below 293 the centroid's pixel, as well as the pixels to the left 294 & right 295 of the centroid's pixel will contain a majority of the dot's value. The further a centroid is away from the exact center of the pixel 295, the more likely that more than the center pixel will have 100% coverage by the dot.

20 Although Fig. 41 only shows centroids differing to the left and below the center, the same relationship obviously holds for centroids above and to the right of center. center. In Case 1, the centroid is exactly in the center of the middle pixel 295. The center pixel 295 is completely covered by the dot, and the pixels above, below, left, and right are also well covered by the dot. In Case 2, the centroid is to the left of the center of the middle pixel 291. The center pixel is still completely covered by the dot, and the pixel 294 to the left of the center is now completely covered by the dot. The pixels above 292 and

below 293 are still well covered. In Case 3, the centroid is below the center of the middle pixel 291. The center pixel 291 is still completely covered by the dot 291, and the pixel below center is now completely covered by the dot.

5 The pixels left 294 and right 295 of center are still well covered. In Case 4, the centroid is left and below the center of the middle pixel. The center pixel 291 is still completely covered by the dot, and both the pixel to the left of center 294 and the pixel below center 293 are
10 completely covered by the dot.

The algorithm for updating the centroid uses the distance of the centroid from the center of the middle pixel 291 in order to select 3 representative pixels and thus decide the value of the dot:

15 Pixel 1: the pixel containing the centroid

Pixel 2: the pixel to the left of Pixel 1 if the centroid's X coordinate (column value) is $< \frac{1}{2}$, otherwise the pixel to the right of Pixel 1.

Pixel 3: the pixel above pixel 1 if the centroid's Y coordinate (row value) is $< \frac{1}{2}$, otherwise the pixel below
20 Pixel 1.

As shown in Fig.42, the value of each pixel is output to a pre-calculated lookup table 301. The 3 pixels are fed into a 12-bit lookup table, which outputs a single bit
25 indicating the value of the dot - on or off. The lookup table 301 is constructed at chip definition time, and can be compiled into about 500 gates. The lookup table can be a simple threshold table, with the exception that the center pixel (Pixel 1) is weighted more heavily.

30 Step 3: Update the centroid Δ s for each row in the column

The idea of the Δ s processing is to use the previous bit history to generate a 'perfect' dot at the expected centroid location for each row in a current column. The
35 actual pixels (from the CCD) are compared with the expected 'perfect' pixels. If the two match, then the actual centroid

location must be exactly in the expected position, so the centroid Δ s must be valid and not need updating. Otherwise a process of changing the centroid Δ s needs to occur in order to best fit the expected centroid location to the actual
5 data. The new centroid Δ s will be used for processing the dot in the next column.

Updating the centroid Δ s is done as a subsequent process from Step 2 for the following reasons:

to reduce complexity in design, so that it can be
10 performed as Step 2 of Phase 1 there is enough bandwidth remaining to allow it to allow reuse of DRAM buffers, and to ensure that all the data required for centroid updating is available at the start of the process without special pipelining.

15 The centroid Δ are processed as Δ_{column} Δ_{row} respectively to reduce complexity.

Although a given dot is 3 pixels in diameter, it is likely to occur in a 4x4 pixel area. However the edge of one dot will as a result be in the same pixel as the edge of the
20 next dot. For this reason, centroid updating requires more than simply the information about a given single dot.

Fig.43 shows a single dot 310 from the previous column with a given centroid 311. In this example, the dot 310 extend Δ over 4 pixel columns 312-315 and in fact, part of
25 the previous dot column's dot (coordinate = (Prevcolumn, Current Row)) has entered the current column for the dot on the current row. If the dot in the current row and column was white, we would expect the rightmost pixel column 314 from the previous dot column to be a low value, since there
30 is only the dot information from the previous column's dot (the current column's dot is white). From this we can see that the higher the pixel value is in this pixel column 315, the more the centroid should be to the right. Of course, if the dot to the right was also black, we cannot adjust the
35 centroid as we cannot get information sub-pixel. The same

can be said for the dots to the left, above and below the dot at dot coordinates (PrevColumn, CurrentRow).

From this we can say that a maximum of 5 pixel columns and rows are required. It is possible to simplify the situation by taking the cases of row and column centroid Δ s separately, treating them as the same problem, only rotated 90 degrees.

Taking the horizontal case first, it is necessary to change the column centroid Δ s if the expected pixels don't match the detected pixels. From the bit history, the value of the bits found for the Current Row in the current dot column, the previous dot column, and the (previous-1)th dot column are known. The expected centroid location is also known. Using these two pieces of information, it is possible to generate a 20 bit expected bit pattern should the read be 'perfect'. The 20 bit bit-pattern represents the expected Δ values for each of the 5 pixels across the horizontal dimension. The first nibble would represent the rightmost pixel of the leftmost dot. The next 3 nibbles represent the 3 pixels across the center of the dot 310 from the previous column, and the last nibble would be the leftmost pixel 317 of the rightmost dot (from the current column).

If the expected centroid is in the center of the pixel, we would expect a 20 bit pattern based on the following table:

Bit history	Expected pixels
000	00000
001	0000D
010	0DFD0
011	0DFDD
100	D0000
101	D000D
110	DDFD0
111	DDFDD

The pixels to the left and right of the center dot are either 0 or D depending on whether the bit was a 0 or 1 respectively. The center three pixels are either 000 or DFD depending on whether the bit was a 0 or 1 respectively.
5 These values are based on the physical area taken by a dot for a given pixel. Depending on the distance of the centroid from the exact center of the pixel, we would expect data shifted slightly, which really only affects the pixels
10 either side of the center pixel. Since there are 16 possibilities, it is possible to divide the distance from the center by 16 and use that amount to shift the expected pixels.

Once the 20 bit 5 pixel expected value has been
15 determined it can be compared against the actual pixels read. This can proceed by subtracting the expected pixels from the actual pixels read on a pixel by pixel basis, and finally adding the differences together to obtain a distance from the expected Δ values.

20 Fig. 44 illustrates one form of implementation of the above algorithm which includes a look up table 320 which receives the bit history 322 and central fractional component 323 and outputs 324 the corresponding 20 bit number which is subtracted 321 from the central pixel input
25 326 to produce a pixel difference 327.

This process is carried out for the expected centroid and once for a shift of the centroid left and right by 1 amount in Δ_{column} . The centroid with the smallest difference from the actual pixels is considered to be the
30 'winner' and the Δ_{column} updated accordingly (which hopefully is 'no change'). As a result, a Δ_{column} cannot change by more than 1 each dot column.

The process is repeated for the vertical pixels, and Δ_{row} is consequentially updated.

35 There is a large amount of scope here for parallelism.

Depending on the rate of the clock chosen for the ACP unit 31 these units can be placed in series (and thus the testing of 3 different Δ could occur in consecutive clock cycles), or in parallel where all 3 can be tested simultaneously. If
5 the clock rate is fast enough, there is less need for parallelism.

Bandwidth utilization

It is necessary to read the old Δ of the Δ s, and to write them out again. This takes 10% of the bandwidth:

10 $2 * (76(3150/32) + 2*3150) = 27,648\text{ns} = 10\% \text{ of bandwidth}$

It is necessary to read the bit history for the given row as we update its Δ s. Each byte contains 2 row's bit histories, thus taking 2.5% of the bandwidth:

$76((3150/2)/32) + 2*(3150/2) = 4,085\text{ns} = 2.5\% \text{ of bandwidth}$

15 In the worst case of pixel drift due to a 1% rotation, centroids will shift 1 column every 57 pixel rows, but since a dot is 3 pixels in diameter, a given pixel column will be valid for 171 pixel rows ($3*57$). As a byte contains 2 pixels, the number of bytes valid in cached reads will be a
20 worst case of 86 (out of 128 read). The worst case timing for 5 columns is therefore 31% bandwidth.

$5 * (((9450/(128 * 2)) * 320) * 128/86) = 88,112\text{ns} = 31\% \text{ of bandwidth.}$

The total bandwidth required for the updating the
25 centroid Δ is summarised in the following table:

Read/Write centroid Δ	10%
Read bit history	2.5%
Read 5 columns of pixel data	31%
TOTAL	43.5%

Memory usage for Phase 2:

The 2MB bit-image DRAM area is read from and written to
30 during Phase 2 processing. The 2MB pixel-data DRAM area is

read.

The 0.5MB scratch DRAM area is used for storing row data, namely:

Centroid array	24bits (16:8) * 2 * 3150 = 18,900 bytes
Bit History array	3 bits * 3150 entries (2 per byte) = 1575 bytes

5

Phase 3 -Unscramble and XOR the raw data

Returning to Fig.36, the next step in decoding is to unscramble and XOR the raw data. The 2MB byte image, as taken from the Artcard, is in a scrambled XORed form. It must be unscrambled and re-XORed to retrieve the bit image necessary for the Reed Solomon decoder in phase 4.

Turning to Fig.45, the unscrambling process 330 takes a 2MB scrambled byte image 331 and writes an unscrambled 2MB image 332. The process cannot reasonably be performed in-place, so 2 sets of 2MB areas are utilised. The scrambled data 331 is in symbol block order arranged in a 16x16 array, with symbol block 0 (334) having all the symbol 0's from all the code words in random order. Symbol block 1 has all the symbol 1's from all the code words in random order etc. Since there are only 255 symbols, the 256th symbol block is currently unused.

A linear feedback shift register is used to determine the relationship between the position within a symbol block eg. 334 and what code word eg. 355 it came from. This works as long as the same seed is used when generating the original Artcard images. The XOR of bytes from alternative source lines with 0xAA and 0x55 respectively is effectively free (in time) since the bottleneck of time is waiting for the DRAM to be ready to read/write to non-sequential addresses.

The timing of the unscrambling XOR process is effectively 2MB of random byte-reads, and 2MB of random

byte-writes i.e. $2 * (2\text{MB} * 76\text{ns} + 2\text{MB} * 2\text{ns}) =$
327,155,712ns or approximately 0.33 seconds. This timing
assumes no caching.

Phase 4 - Reed Solomon decode

5 This phase is a loop, iterating through copies of the
data in the bit image, passing them to the Reed-Solomon
decode module until either a successful decode is made or
until there are no more copies to attempt decode from.

 The Reed-Solomon decoder used can be the VLIW
10 processor, suitably programmed or, alternatively, a separate
hardwired core such as LSI Logic's L64712. The L64712 has a
throughput of 50Mbits per second (around 6.25MB per second),
so the time may be bound by the speed of the Reed-Solomon
decoder rather than the 2MB read and 1 MB write memory
15 access time (500MB/sec for sequential accesses). The time
taken in the worst case is thus $2/6.25\text{s} =$ approximately 0.32
seconds.

Phase 5 Running the Vark script

 The overall time taken to read the Artcard 9 and decode
20 it is therefore approximately 2.15 seconds. The apparent
delay to the user is actually only 0.65 seconds (the total
of Phases 3 and 4), since the Artcard stops moving after 1.5
seconds.

 Once the Artcard is loaded, the Artvark script must be
25 interpreted. Rather than run the script immediately, the
script is only run upon the pressing of the 'Print' button
13 (Fig.1). The taken to run the script will vary depending
on the complexity of the script, and must be taken into
account for the perceived delay between pressing the print
30 button and the actual print button and the actual printing.

 As noted previously, the VLIW processor 74 is a digital
processing system that accelerates computationally expensive
Vark functions. The balance of functions performed in
software by the CPU core 72, and in hardware by the VLIW
35 processor 74 will be implementation dependent. The goal of
the VLIW processor 74 is to assist all Artcard styles to

execute in a time that does not seem too slow to the user.

As CPUs become faster and more powerful, the number of functions requiring hardware acceleration becomes less and

less. The VLIW processor has a microcoded ALU sub-system

5 that allows general hardware speed up of the following time-critical functions.

1) Image access mechanisms for general software processing

2) Image convolver.

3) Data driven image warper

10 4) Image scaling

5) Image tessellation

6) Affine transform

7) Image compositor

8) Color space transform

15 9) Histogram collector

10) Illumination of the Image

11) Brush stamper

12) Histogram collector

13) CCD image to internal image conversion

20 14) Construction of image pyramids (used by warper & for brushing)

The following table summarizes the time taken for each Vark operation if implemented in the ALU model. The method of implementing the function using the ALU model is

25 described hereinafter.

Operation	Speed of Operation	1500 * 1000 image	
		1 channel	3 channels
Image composite	1 cycle per output pixel	0.015 s	0.045 s
Image convolve	k/3 cycles per output pixel		
	(k = kernel size)	0.045 s	0.135 s
	3x3 convolve	0.125 s	0.375 s

	5x5 convolve 7x7 convolve	0.245 s	0.735 s
Image warp	8 cycles per pixel	0.120 s	0.360 s
Histogram collect	2 cycles per pixel	0.030 s	0.090 s
Image Tessellate	1/3 cycle per pixel	0.005 s	0.015 s
Image sub-pixel Translate	1 cycle per output pixel	-	-
Color lookup replace	½ cycle per pixel	0.008 s	0.023
Color space transform	8 cycles per pixel	0.120 s	0.360 s
Convert CCD image to internal image (including color convert & scale)	4 cycles per output pixel	0.06 s	0.18 s
Construct image pyramid	1 cycle per input pixel	0.015 s	0.045 s
Scale	Maximum of: 2 cycles per input pixel 2 cycles per output pixel 2 cycles per output pixel (scaled in X only)	0.015 s (minimum)	0.045 s (minimum)
Affine transform	2 cycles per output pixel	0.03 s	0.09 s
Brush rotate/translat	?		

e and composite			
Tile Image	4-8 cycles per output pixel	0.015 s to 0.030 s	0.060 s to 0.120 s to for 4 channels (Lab, texture)
Illuminate image	Cycles per pixel	0.008 s	0.023 s
Ambient only	½	0.015 s	0.045 s
Directional light	1 6	0.09 s 0.09 s	0.27 s 0.27 s
Directional (bm)	6 9	0.137 s 0.137 s	0.41 s 0.41 s
Omni light	9	0.18 s	0.54 s
Omni (bm)	12		
Spotlight			
Spotlight (bm)			
(bm) = bumpmap			

For example, to convert a CCD image, collect histogram
& perform lookup-color replacement (for image enhancement)
takes: 9+2+0.5 cycles per pixel, or 11.5 cycles. For a 1500
5 x 1000 image that is 172,500,000, or approximately 0.2
seconds per component, or 0.6 seconds for all 3 components.
Add a simple warp, and the total comes to 0.6 + 0.36, almost
1 second.

Image Convolver

10 A convolve is a weighted average around a center pixel.
The average may be a simple sum, a sum of absolute values,
the absolute value of a sum, or sums truncated at 0.

The image convolver is a general-purpose convolver,
allowing a variety of functions to be implemented by varying
15 the values within a variable-sized coefficient kernel. The

kernel sizes supported are 3x3, 5x5 and 7x7 only.

Turning now to Fig.47, there is illustrated 340 an example of the convolution process. The pixel component values fed into the convolver process 341 come from a Box Read Iterator 342. The Iterator 342 provides the image data row by row, and within each row, pixel by pixel. The output from the convolver 341 is sent to a Sequential Write Iterator 344, which stores the resultant image in a valid image format.

10 A Coefficient Kernel 346 is a lookup table in DRAM. The kernel is arranged with coefficients in the same order as the Box Read Iterator 342. Each coefficient entry is 8 bits. A simple Sequential Read Iterator can be used to index into the kernel 346 and thus provide the coefficients. It
15 simulates an image with ImageWidth equal to the kernel size, and a Loop option is set so that the kernel would continuously be provided.

One form of implementation of the convolve process on an ALU unit is as illustrated in Fig. 46. The following
20 constants are set by software:

Constant	Value
K ₁	Kernel size (9, 25, or 49)

The control logic is used to count down the number of multiply/adds per pixel. When the count (accumulated in Latch₂) reaches 0, the control signal generated is used to
25 write out the current convolve value (from Latch₁) and to reset the count. In this way, one control logic block can be used for a number of parallel convolve streams.

Each cycle the multiply ALU can perform one multiply/add to incorporate the appropriate part of a pixel.
30 The number of cycles taken to sum up all the values is therefore the number of entries in the kernel. Since this is compute bound, it is appropriate to divide the image into multiple sections and process them in parallel on different ALU units.

On a 7x7 kernel, the time taken for each pixel is 49 cycles, or 490ns. Since each cache line holds 32 pixels, the time available for memory access is 12,740ns. $((32-7+1) \times 490\text{ns})$. The time taken to read 7 cache lines and write 1 is worse case 1,120ns $(8 \times 140\text{ns})$, all accesses to same DRAM bank). Consequently it is possible to process up to 10 pixels in parallel given unlimited resources. Given a limited number of ALUs it is possible to do at best 4 in parallel. The time taken to therefore perform the convolution using a 7x7 kernel is 0.18375 seconds $(1500 \times 1000 \times 490\text{ns} / 4 = 183,750,000\text{ns})$.

On a 5x5 kernel, the time taken for each pixel is 25 cycles, or 250ns. Since each cache line holds 32 pixels, the time available for memory access is 7,000ns. $((32-5+1) \times 250\text{ns})$. The time taken to read 5 cache lines and write 1 is worse case 840ns $(6 \times 140\text{ns})$, all accesses to same DRAM bank). Consequently it is possible to process up to 7 pixels in parallel given unlimited resources. Given a limited number of ALUs it is possible to do at best 4. The time taken to therefore perform the convolution using a 5x5 kernel is 0.09375 seconds $(1500 \times 1000 \times 250\text{ns} / 4 = 93,750,000\text{ns})$.

On a 3x3 kernel, the time taken for each pixel is 9 cycles, or 90ns. Since each cache line holds 32 pixels, the time available for memory access is 2,700ns. $((32-3+1) \times 90\text{ns})$. The time taken to read 3 cache lines and write 1 is worse case 560ns $(4 \times 140\text{ns})$, all accesses to same DRAM bank). Consequently it is possible to process up to 4 pixels in parallel given unlimited resources. Given a limited number of ALUs and Read/Write Iterators it is possible to do at best 4. The time taken to therefore perform the convolution using a 3x3 kernel is 0.03375 seconds $(1500 \times 1000 \times 90\text{ns} / 4 = 33,750,000\text{ns})$. Consequently each output pixel takes $\text{kernel size}/3$ cycles to compute. The actual timings are summarised in the following table:

Kernel size	Time taken to calculate output pixel	Time to process 1 channel at 1500x1000	Time to Process 3 channels at 1500x1000
3x3 (9)	3 cycles	0.045 seconds	0.135 seconds
5x5 (25)	8 1/3 cycles	0.125 seconds	0.375 seconds
7x7 (49)	16 1/3 cycles	0.245 seconds	0.735 seconds

Image Compositor

Compositing is to add a foreground image to a background image using a matte or a channel to govern the appropriate proportions of background and foreground in the final image. Two styles of compositing are preferably supported, regular compositing and associated compositing.

The rules for the two styles are:

Regular composite: $\text{new Value} = \text{Foreground} + (\text{Background} - \text{Foreground}) \alpha$

Associated composite: $\text{new value} = \text{Foreground} + (1 - \alpha) \text{Background}$

The difference then, is that with associated compositing, the foreground has been pre-multiplied with the matte, while in regular compositing it has not. An example of the compositing process is as illustrated in Fig. 48.

The alpha channel has values from 0 to 255 corresponding to the range 0 to 1.

Regular Composite

A regular composite is implemented as:

$$\text{Foreground} + (\text{Background} - \text{Foreground}) * \alpha / 255$$

The division by $X/255$ is approximated by $257X/65536$. An implementation of the compositing process is shown in more detail in Fig. 49, where the following constant is set by software:

Constant	Value
K_1	257

Since 4 Iterators are required, the composite process takes 1 cycle per pixel, with a utilization of only half of the ALUs. The composite process is only run on a single channel. To composite a 3-channel image with another, the
5 compositor must be run 3 times, once for each channel.

The time taken to composite a full size single channel is 0.015s ($1500 * 1000 * 1 * 10\text{ns}$), or 0.045s to composite all 3 channels.

To approximate a divide by 255 it is possible to
10 multiply by 257 and then divide by 65536. It can also be achieved by a single add ($256 * x + x$) and ignoring (except for rounding purposes) the final 16 bits of the result.

As shown in Fig. 41, the compositor process requires 3 Sequential Read Iterators 351-353 and 1 Sequential Write
15 Iterator 355, and is implemented as microcode using a Adder ALU in conjunction with a multiplier ALU. Composite time is 1 cycle (10ns) per-pixel. Different microcode is required for associated and regular compositing, although the average time per pixel composite is the same.

20 The composite process is only run on a single channel. To composite one 3-channel image with another, the compositor must be run 3 times, once for each channel. As the a channel is the same for each composite, it must be read each time. However it should be noted that to transfer
25 (read or write) 4 x 32 byte cache-lines in the best case takes 320ns. The pipeline gives an average of 1 cycle per pixel composite, taking 32 cycles or 320ns (at 100MHz) to composite the 32 pixels, so the a channel is effectively read for free. An entire channel can therefore be composited
30 in:

$$1500/32 * 1000 * 320\text{ns} = 15,040,000\text{ns} = 0.015\text{seconds}.$$

The time taken to composite a full size 3 channel image is therefore 0.045 seconds.

Construct Image Pyramid

35 Several functions, such as warping, tiling and brushing, require the average value of a given area of

pixels. Rather than calculate the value for each area given, these functions preferably make use of an image pyramid. As illustrated previously in Fig.7, an image pyramid 360 is effectively a multi-resolution pixelmap. The original image is a 1:1 representation. Sub-sampling by 2:1 in each dimension produces an image $\frac{1}{4}$ the original size. This process continues until the entire image is represented by a single pixel.

An image pyramid is constructed from an original image, and consumes $\frac{1}{3}$ of the size taken up by the original image ($\frac{1}{4} + \frac{1}{16} + \frac{1}{64} + \dots$). For an original image of 1500 x 1000 the corresponding image pyramid is approximately $\frac{1}{3}$ MB

The image pyramid can be constructed via a 3x3 convolve performed on 1 in 4 input image pixels advancing the center of the convolve kernel by 2 pixels each dimension. A 3x3 convolve results in higher accuracy than simply averaging 4 pixels, and has the added advantage that coordinates on different pyramid levels differ only by shifting 1 bit per level.

The construction of an entire pyramid relies on a software loop that calls the pyramid level construction function once for each level of the pyramid.

The timing to produce 1 level of the pyramid is $\frac{9}{4} * \frac{1}{4}$ of the resolution of the input image since we are generating an image $\frac{1}{4}$ of the size of the original. Thus for a 1500 x 1000 image:

Timing to produce level 1 of pyramid = $\frac{9}{4} * 750 * 500$
= 843, 750 cycles

Timing to produce level 2 of pyramid = $\frac{9}{4} * 375 * 250$
= 210, 938 cycles

Timing to produce level 3 of pyramid = $\frac{9}{4} * 188 * 125$
= 52, 735 cycles

Etc.

The total time is $\frac{3}{4}$ cycle per original image pixel (image pyramid is $\frac{1}{3}$ of original image size, and each pixel takes $\frac{9}{4}$ cycles to be calculated, i.e. $\frac{1}{3} * \frac{9}{4} = \frac{3}{4}$). In

the case of a 1500 x 1000 image is 1,125,000 cycles (at 100MHz), or 0.011 seconds. This timing is for a single color channel, 3 color channels require 0.034 seconds processing time.

5 General Data Driven Image Warper

The ACP 31 is able to carry out image warping manipulations of the input image. The principles of image warping are well-known in theory. One thorough text book reference on the process of warping is "Digital Image
10 Warping" by George Wolberg published in 1990 by the IEEE Computer Society Press, Los Alamitos, California. The warping process utilizes a warp map which forms part of the data fed in via Artcard 9. The warp map can be arbitrarily dimensioned in accordance with requirements and provides
15 information of a mapping of input pixels to output pixels. Unfortunately, the utilization of arbitrarily sized warp maps presents a number of problems which must be solved by the image warper.

Turning to Fig 50, a warp map 365, having dimensions
20 AxB comprises array values of a certain magnitude (for example 8 bit values from 0 - 255) which set out the coordinate of a theoretical input image which maps to the corresponding "theoretical" output image having the same array coordinate indices. Unfortunately, any output image
25 eg. 366 will have its own dimensions CxD which may further be totally different from an input image which may have its own dimensions, ExF. Hence, it is necessary to facilitate the remapping of the warp map 365 so that it can be utilised for output image 366 to determine, for each output pixel,
30 the corresponding area or region of the input image 367 from which the output pixel color data is to be constructed. For each output pixel in output image 366 it is necessary to first determine a corresponding warp map value from warp map 365. This may include the need to bilinearly interpolate
35 the surrounding warp map values when an output image pixel maps to a fractional position within warp map table 365.

The result of this process will give the location of an input image pixel in a "theoretical" image which will be dimensioned by the size of each data value within the warp map 365. These values must be re-scaled so as to map the
5 theoretical image to the corresponding actual input image 367.

In order to determine the actual value and output image pixel should take so as to avoid aliasing effects, adjacent output image pixels should be examined to determine a region
10 of input image pixels 367 which will contribute to the final output image pixel value. In this respect, the image pyramid is utilised as will become more apparent hereinafter.

The image warper performs several tasks in order to
15 warp an image.

- Scale the warp map to match the output image size.
- Determine the span of the region of input image pixels represented in each output pixel.
- Calculate the final output pixel value via tri-
20 linear interpolation from the input image pyramid

Scale warp map

As noted previously, in a data driven warp, there is the need for a warp map that describes, for each output pixel, the center of a corresponding input image map.
25 Instead of having a single warp map as previously described, containing interleaved x and y value information, it is possible to treat the X and Y coordinates as separate channels.

Consequently, preferably there are two warp maps: an X
30 warp map showing the warping of X coordinates, and a Y warp map, showing the warping of the Y coordinates. As noted previously, the warp map 365 can have a different spatial resolution than the image they being scaled (for example a 32 x 32 warp-map 365 may adequately describe a warp for a
35 1500 x 1000 image 366). In addition, the warp maps can be represented by 8 or 16 bit values that correspond to the

size of the image being warped.

There are several steps involved in producing points in the input image space from a given warp map:

1. Determining the corresponding position in the warp map for the output pixel
2. Fetch the values from the warp map for the next step (this can require scaling in the resolution domain if the warp map is only 8 bit values)
3. Bi-linear interpolation of the warp map to determine the actual value
4. Scaling the value to correspond to the input image domain

The first step can be accomplished by multiplying the current X/Y coordinate in the output image by a scale factor (which can be different in X & Y). For example, if the output image was 1500 x 1000, and the warp map was 150 x 100, we scale both X & Y by 1/10.

Fetching the values from the warp map requires access to 2 Lookup tables. One Lookup table indexes into the X warp-map, and the other indexes into the Y warp-map. The lookup table either reads 8 or 16 bit entries from the lookup table, but always returns 16 bit values (clearing the high 8 bits if the original values are only 8 bits).

The next step in the pipeline is to bi-linearly interpolate the looked-up warp map values.

Finally the result from the bi-linear interpolation is scaled to place it in the same domain as the image to be warped. Thus, if the warp map range was 0-255, we scale X by 1500/255, and Y by 1000/255.

The interpolation process is as illustrated in Fig. 51 with the following constants set by software:

Constant	Value
K ₁	Xscale (scales 0-ImageWidth to 0-WarpmapWidth)
K ₂	Yscale (scales 0-ImageHeight to 0-WarpmapHeight)
K ₃	XrangeScale (scales warpmap range (eg 0-255) to

	0-ImageWidth)
K ₄	YrangeScale (scales warpmap range (eg 0-255) to 0-ImageHeight)

The following lookup table is used:

Lookup	Size	Details
LU ₁ and LU ₂	WarpmapWidth x WarpmapHeight	Warpmap lookup. Given [X,Y] the 4 entries required for bi-linear interpolation are returned. Even if entries are only 8 bit, they are returned as 16 bit (high 8 bits 0). Transfer time is 4 entries at 2 bytes per entry. Total time is 8 cycles as 2 lookups are used.

Span calculation

5 The points from the warp map 365 locate centers of
pixel regions in the input image 367. The distance between
input image pixels of adjacent output image pixels will
indicate the size of the regions, and this distance can be
approximated via a span calculation.

10 Turning to Fig.52, for a given current point in the
warp map P1, the previous point on the same line is called
P0, and the previous line's point at the same position is
called P2. We determine the absolute distance in X & Y
between P1 and P0, and between P1 and P2. The maximum
distance in X or Y becomes the span which will be a square
15 approximation of the actual shape.

20 Preferably, the points are processed in a vertical
strip output order, P0 is the previous point on the same
line within a strip, and when P1 is the first point on line
within a strip, then P0 refers to the last point in the
previous strip's corresponding line. P2 is the previous
line's point in the same strip, so it can be kept in a 32-
entry history buffer. The basic of the calculate span

process are as illustrated in Fig. 53 with the details of the process as illustrated in Fig. 54.

The following DRAM FIFO is used:

Lookup	Size	Details
FIFO ₁	8 ImageWidth bytes. [ImageWidth x 2 entries at 32 bits per entry]	P2 history/lookup (both X & Y in same FIFO) P1 is put into the FIFO and taken out again at the same pixel on the following row as P2. Transfer time is 4 cycles (2 x 32 bits, with 1 cycle per 16 bits)

- 5 Since a 32 bit precision span history is kept, in the case of a 1500 pixel wide image being warped 12,000 bytes temporary storage is required.

- 10 Calculation of the span 364 uses 2 Adder ALUs (1 for span calculation, 1 for looping and counting for P0 and P2 histories) takes 7 cycles as follows:

Cycle	Action
1	A = ABS(P1 _x - P2 _x) Store P1 _x in P2 _x history
2	B = ABS(P1 _x - P0 _x) Store P1 _x in P0 _x history
3	A = MAX(A, B)
4	B = ABS(P1 _y - P2 _y) Store P1 _y in P2 _y history
5	A = MAX(A, B)
6	B = ABS(P1 _y - P0 _y) Store P1 _y in P0 _y history
7	A = MAX(A, B)

The history buffers 365, 366 are cached DRAM. The 'Previous Line' (for P2 history) buffer 366 is 32 entries of

span-precision. The 'Previous Point' (for P0 history).
Buffer 365 requires 1 register that is used most of the time
(for calculation of points 1 to 31 of a line in a strip),
and a DRAM buffered set of history values to be used in the
5 calculation of point 0 in a strip's line.

32 bit precision in span history requires 4 cache lines
to hold P2 history, and 2 for P0 history. P0's history is
only written and read out once every 8 lines of 32 pixels to
a temporary storage space of (ImageHeight*4) bytes. Thus a
10 1500 pixel high image being warped requires 6000 bytes
temporary storage, and a total of 6 cache lines.

Tri-linear interpolation

Having determined the center and span of the area from
the input image to be averaged, the final part of the warp
15 process is to determine the value of the output pixel. Since
a single output pixel could theoretically be represented by
the entire input image, it is potentially too time-consuming
to actually read and average the specific area of the input
image contributing to the output pixel. Instead, it is
20 possible to approximate the pixel value by using an image
pyramid of the input image.

If the span is 1 or less, it is necessary only to read
the original image's pixels around the given coordinate, and
perform bi-linear interpolation. If the span is greater than
25 1, we must read two appropriate levels of the image pyramid
and perform tri-linear interpolation. Performing linear
interpolation between two levels of the image pyramid is not
strictly correct, but gives acceptable results (it errs on
the side of blurring the resultant image).

30 Turning to Fig.55, generally speaking, for a given span
's', it is necessary to read image pyramid levels given by
 $\ln_2 s$ (370) and $\ln_2 s + 1$ (371). $\ln_2 s$ is simply decoding the
highest set bit of s. We must bi-linear interpolate to
determine the value for the pixel value on each of the two
35 levels 370, 371 of the pyramid, and then interpolate between
levels.

As shown in Fig.56, it is necessary to first interpolate in X and Y for each pyramid level before interpolating between the pyramid levels to obtain a final output value 373.

5 The image pyramid address mode issued to generate addresses for pixel coordinates at (x, y) on pyramid level s & s+1. Each level of the image pyramid contains pixels sequential in x. Hence, reads in x are likely to be cache hits.

10 Reasonable cache coherence can be obtained as local regions in the output image are typically locally coherent in the input image (perhaps at a different scale however, but coherent within the scale). Since it is not possible to know the relationship between the input and output images,
15 we ensure that output pixels are written in a vertical strip (via a Vertical-Strip Iterator) in order to best make use of cache coherence.

Tri-linear interpolation can be completed in as few as 2 cycles on average using 4 multiply ALUs and all 4 adder
20 ALUs as a pipeline and assuming no memory access required. But since all the interpolation values are derived from the image pyramids, interpolation speed is completely dependent on cache coherence (not to mention the other units are busy doing warp-map scaling and span calculations). As many cache
25 lines as possible should therefore be available to the image-pyramid reading. The best speed will be 8 cycles, using 2 Multiply ALUs.

The output pixels are written out to the DRAM via a Vertical-Strip Write Iterator that uses 2 cache lines. The
30 speed is therefore limited to a minimum of 8 cycles per output pixel. If the scaling of the warp map requires 8 or fewer cycles, then the overall speed will be unchanged. Otherwise the throughput is the time taken to scale the warp map. In most cases the warp map will be scaled up to match
35 the size of the photo.

Assuming a warp map that requires 8 or fewer cycles per

pixel to scale, the time taken to convert a single color component of image is therefore 0.12s (1500 * 1000 * 8 cycles * 10ns per cycle).

Histogram Collector

5 The histogram collector is a microcode program that takes an image channel as input, and produces a histogram as output. Each of a channel's pixels has a value in the range 0-255. Consequently there are 256 entries in the histogram table, each entry 32 bits - large enough to contain a count
10 of an entire 1500x1000 image.

As shown in Fig.57, since the histogram represents a summary of the entire image, a Sequential Read Iterator 378 is sufficient for the input. The histogram itself can be completely cached, requiring 32 cache lines (1K).

15 The microcode has two passes: an initialization pass which sets all the counts to zero, and then a "count" stage that increments the appropriate counter for each pixel read from the image. The first stage requires the Address Unit and a single Adder ALU, with the address of the histogram
20 table 377 for initialising.

Relative Microcode Address	Address Unit A = Base address of histogram	Adder Unit 1
0	Write 0 to A + (Adder1.Out1 << 2)	Out1 = A A = A - 1 BNZ 0
1	Rest of processing	Rest of processing

The second stage processes the actual pixels from the image, and uses 4 Adder ALUs:

25

	Adder 1	Adder 2	Adder 3	Adder 4	Address Unit
1	A = 0			A = -1	

2	Out1 = BZ A 2 A = pixel	A = Adder1.Out1 Z = pixel - Adder1.Out1	A = Adr.Out1	A = A + 1	Out1 = Read 4 bytes from: (A + (Adder1.Out1 << 2))
3		Out1 = A	Out1 = A	Out1 = A A = Adder3.Ou t1	Write Adder4.Out1 to: (A + (Adder 2.Out << 2)
4					Write Adder4.Out1 to: (A + (Adder 2.Out << 2) Flush caches

The Zero flag from Adder2 cycle 2 is used to stay at microcode address 2 for as long as the input pixel is the same. When it changes, the new count is written out in
5 microcode address 3, and processing resumes at microcode address 2. Microcode address 4 is used at the end, when there are no more pixels to be read.

Stage 1 takes 256 cycles, or 2560ns. Stage 2 varies according to the values of the pixels. The worst case time
10 for lookup table replacement is 2 cycles per image pixel if every pixel is not the same as its neighbor. The time taken for a single color lookup is 0.03s (1500 x 1000 x 2 cycle per pixel x 10ns per cycle = 30,000,000ns). The time taken for 3 color components is 3 times this amount, or 0.09s.

15 Color Transform

Color transformation is achieved in two main ways:

Lookup table replacement

Color space conversion

Lookup Table Replacement

20 As illustrated in Fig.51, one of the simplest ways to transform the color of a pixel is to encode an arbitrarily complex transform function into a lookup table 380. The component color value of the pixel is used to lookup 381 the new component value of the pixel. For each pixel read from a

Sequential Read Iterator, its new value is read from the New Color Table 380, and written to a Sequential Write Iterator 383. The input image can be processed simultaneously in two halves to make effective use of memory bandwidth. The

5 following lookup table is used:

Lookup	Size	Details
LU ₁	256 entries 8 bits per entry	Replacement[X] Table indexed by the 8 highest significant bits of X. Resultant 8 bits treated as fixed point 0:8

The total process requires 2 Sequential Read Iterators and 2 Sequential Write iterators. The 2 New Color Tables require 8 cache lines each to hold the 256 bytes (256
10 entries of 1 byte).

The average time for lookup table replacement is therefore $\frac{1}{2}$ cycle per image pixel. The time taken for a single color lookup is 0.0075s ($1500 \times 1000 \times \frac{1}{2}$ cycle per pixel $\times 10$ ns per cycle = 7,500,000ns). The time taken for 3
15 color components is 3 times this amount, or 0.0225s. Each color component has to be processed one after the other under control of software.

Color Space Conversion

Color Space conversion is only required when moving
20 between color spaces. The CCD images are captured in RGB color space, and printing occurs in CMY color space, while clients of the ACP 31 likely process images in the Lab color space. All of the input color space channels are typically required as input to determine each output channel's
25 component value. Thus the logical process is as illustrated 385 in Fig.59.

Simply, conversion between Lab, RGB, and CMY is fairly straightforward. However the individual color profile of a particular device can vary considerably. Consequently, to
30 allow future CCDs, inks, and printers, the ACP 31 performs

color space conversion by means of tri-linear interpolation from color space conversion lookup tables.

Color coherence tends to be area based rather than line based. To aid cache coherence during tri-linear

5 interpolation lookups, it is best to process an image in vertical strips. Thus the read 386-388 and write 389 iterators would be Vertical-Strip Iterators.

Tri-linear color space conversion

For each output color component, a single 3D table
10 mapping the input color space to the output color component is required. For example, to convert CCD images from RGB to Lab, 3 tables calibrated to the physical characteristics of the CCD are required:

RGB->L

15 RGB->a

RGB->b

To convert from Lab to CMY, 3 tables calibrated to the physical characteristics of the ink/prINTER are required:

Lab->C

20 Lab->M

Lab->Y

The 8-bit input color components are treated as fixed-point numbers (3:5) in order to index into the conversion tables. The 3 bits of integer give the index, and the 5 bits
25 of fraction are used for interpolation. Since 3 bits gives 8 values, 3 dimensions gives 512 entries (8 x 8 x 8). The size of each entry is 1 byte, requiring 512 bytes per table.

The Convert Color Space process can therefore be implemented as shown in Fig. 60 and the following lookup
30 table is used:

Lookup	Size	Details
LU ₁	8 x 8 x 8 entries 512 entries 8 bits per entry	Convert[X, Y, Z] Table indexed by the 3 highest bits of X, Y, and Z. 8 entries returned from Tri-linear index address unit

		Resultant 8 bits treated as fixed point 8:0 Transfer time is 8 entries at 1 byte per entry
--	--	---

Tri-linear interpolation returns interpolation between 8 values. Each 8 bit value takes 1 cycle to be returned from the lookup, for a total of 8 cycles. The tri-linear interpolation also takes 8 cycles when 2 Multiply ALUs are
5 used per cycle. General tri-linear interpolation information is given in the ALU section of this document. The 512 bytes for the lookup table fits in 16 cache lines.

The time taken to convert a single color component of image is therefore 0.105s ($1500 * 1000 * 7 \text{ cycles} * 10\text{ns per cycle}$). To convert 3 components takes 0.415s. Fortunately,
10 the color space conversion for printout takes place on the fly during printout itself, so is not a perceived delay.

If color components are converted separately, they must not overwrite their input color space components since all
15 color components from the input color space are required for converting each component.

Since only 1 multiply unit is used to perform the interpolation, it is alternatively possible to do the entire Lab->CMY conversion as a single pass. This would require 3
20 Vertical-Strip Read Iterators, 3 Vertical-Strip Write Iterators, and access to 3 conversion tables simultaneously. In that case, it is possible to write back onto the input image and thus use no extra memory. However, access to 3 conversion tables equals 1/3 of the caching for each, that
25 could lead to high latency for the overall process.

Affine Transform

Prior to compositing an image with a photo, it may be necessary to rotate, scale and translate it. If the image is only being translated, it can be faster to use a direct sub-
30 pixel translation function. However, rotation, scale-up and translation can all be incorporated into a single affine transform.

A general affine transform can be included as an accelerated function. Affine transforms are limited to 2D, and if scaling down, input images should be pre-scaled via the Scale function. Having a general affine transform function allows an output image to be constructed one block at a time, and can reduce the time taken to perform a number of transformations on an image since all can be applied at the same time.

A transformation matrix needs to be supplied by the client - the matrix should be the inverse matrix of the transformation desired i.e. applying the matrix to the output pixel coordinate will give the input coordinate.

A 2D matrix is usually represented as a 3 x 3 array:

$$\begin{bmatrix} a & b & 0 \\ c & d & 0 \\ e & f & 1 \end{bmatrix}$$

Since the 3rd column is always [0, 0, 1] clients do not need to specify it. Clients instead specify a, b, c, d, e, and f.

Given a coordinate in the output image (x, y) whose top left pixel coordinate is given as (0, 0), the input coordinate is specified by: (ax + cy + e, bx + dy + f). Once the input coordinate is determined, the input image is sampled to arrive at the pixel value. Bi-linear interpolation of input image pixels is used to determine the value of the pixel at the calculated coordinate. Since affine transforms preserve parallel lines, images are processed in output vertical strips of 32 pixels wide for best average input image cache coherence.

Three Multiply ALUs are required to perform the bi-linear interpolation in 2 cycles. Multiply ALUs 1 and 2 do linear interpolation in X for lines Y and Y+1 respectively, and Multiply ALU 3 does linear interpolation in Y between the values output by Multiply ALUs 1 and 2.

As we move to the right across an output line in X, 2

Adder ALUs calculate the actual input image coordinates by adding 'a' to the current X value, and 'b' to the current Y value respectively. When we advance to the next line (either the next line in a vertical strip after processing a maximum of 32 pixels, or to the first line in a new vertical strip) we update X and Y to pre-calculated start coordinate values constants for the given block

The process for calculating an input coordinate is given in Fig. 61 where the following constants are set by software:

Calculate Pixel

Once we have the input image coordinates, the input image must be sampled. A lookup table is used to return the values at the specified coordinates in readiness for bilinear interpolation. The basic process is as indicated in Fig. 62 and the following lookup table is used:

Lookup	Size	Details
LU ₁	Image width by Image height 8 bits per entry	Bilinear Image lookup [X, Y] Table indexed by the integer part of X and Y. 4 entries returned from Bilinear index address unit, 2 per cycle. Each 8 bit entry treated as fixed point 8:0 Transfer time is 2 cycles (2 16 bit entries in FIFO hold the 4 8 bit entries)

The affine transform requires all 4 Multiply Units and all 4 Adder ALUs, and with good cache coherence can perform an affine transform with an average of 2 cycles per output pixel. This timing assumes good cache coherence, which is true for non-skewed images. Worst case timings are severely skewed images, which meaningful Vark scripts are unlikely to contain.

The time taken to transform a 128 x 128 image is therefore 0.00033 seconds (32,768 cycles). If this is a clip

image with 4 channels (including a channel), the total time taken is 0.00131 seconds (131,072 cycles).

A Vertical-Strip Write Iterator is required to output the pixels. No Read Iterator is required. However, since the affine transform accelerator is bound by time taken to access input image pixels, as many cache lines as possible should be allocated to the read of pixels from the input image. At least 32 should be available, and preferably 64 or more.

10 Scaling

Scaling is essentially a re-sampling of an image. Scale up of an image can be performed using the Affine Transform function. Generalized scaling of an image, including scale down, is performed by the hardware accelerated Scale function. Scaling is performed independently in X and Y, so different scale factors can be used in each dimension.

The generalized scale unit must match the Affine Transform scale function in terms of registration. The generalized scaling process is as illustrated in Fig. 63.

20 The scale in X is accomplished by Fant's re-sampling algorithm as illustrated in Fig. 64.

Where the following constants are set by software:

Constant	Value
K_1	Number of input pixels that contribute to an output pixel in X
K_2	$1/K_1$

The following registers are used to hold temporary variables:

25

Variable	Value
Latch ₁	Amount of input pixel remaining unused (starts at 1 and decrements)
Latch ₂	Amount of input pixels remaining to contribute to current output pixel (starts at K_1 and decrements)

Latch ₃	Next pixel (in X)
Latch ₄	Current pixel
Latch ₅	Accumulator for output pixel (unscaled)
Latch ₆	Pixel Scaled in X (output)

The Scale in Y process is illustrated in Fig. 65 and is also accomplished by a slightly altered version of Fant's re-sampling algorithm to account for processing in order of X pixels.

Where the following constants are set by software:

Constant	Value
K ₁	Number of input pixels that contribute to an output pixel in Y
K ₂	1/K ₁

The following registers are used to hold temporary variables:

Variable	Value
Latch ₁	Amount of input pixel remaining unused (starts at 1 and decrements)
Latch ₂	Amount of input pixels remaining to contribute to current output pixel (starts at K ₁ and decrements)
Latch ₃	Next pixel (in Y)
Latch ₄	Current pixel
Latch ₅	Pixel Scaled in Y (output)

The following DRAM FIFOs are used:

Lookup	Size	Details
FIFO ₁	ImageWidth _{OUT} entries 8 bits per entry	1 row of image pixels already scaled in X 1 cycle transfer time
FIFO ₂	ImageWidth _{OUT} entries 16 bits per entry	1 row of image pixels already scaled in X 2 cycles transfer time (1

		byte per cycle)
--	--	-----------------

Tessellate Image

Tessellation of an image is a form of tiling. It involves copying a specially designed "tile" multiple times horizontally and vertically into a second (usually larger) image space. When tessellated, the small tile forms a seamless picture. One example of this is a small tile of a section of a brick wall. It is designed so that when tessellated, it forms a full brick wall. Note that there is no scaling or sub-pixel translation involved in tessellation.

The most cache-coherent way to perform tessellation is to output the image sequentially line by line, and to repeat the same line of the input image for the duration of the line. When we finish the line, the input image must also advance to the next line (and repeat it multiple times across the output line).

An overview of the tessellation function is illustrated in Fig. 66. The Sequential Read Iterator is set up to continuously read a single line of the input tile (StartLine would be 0 and EndLine would be 1). Each input pixel is written to all 3 of the Write Iterators 393-395. A counter 397 in an Adder ALU counts down the number of pixels in an output line, terminating the sequence at the end of the line.

At the end of processing a line, a small software routine updates the Sequential Read Iterator's StartLine and EndLine registers before restarting the microcode and the Sequential Read Iterator (which clears the FIFO and repeats line 2 of the tile). The Write Iterators 393-395 are not updated, and simply keep on writing out to their respective parts of the output image. The net effect is that the tile has one line repeated across an output line, and then the tile is repeated vertically too.

This process does not fully use the memory bandwidth

since we get good cache coherence in the input image, but it does allow the tessellation to function with tiles of any size. The process uses 1 Adder ALU. If the 3 Write Iterators 393-395 each write to 1/3 of the image (breaking the image on tile sized boundaries), then the entire tessellation process takes place at an average speed of 1/3 cycle per output image pixel. For an image of 1500 x 1000, this equates to .005 seconds (5,000,000ns).

Sub-pixel Translator

Before compositing an image with a background, it may be necessary to translate it by a sub-pixel amount in both X and Y. Sub-pixel transforms can increase an image's size by 1 pixel in each dimension. The value of the region outside the image can be client determined, such as a constant value (e.g. black), or edge pixel replication. Typically it will be better to use black.

The sub-pixel translation process is as illustrated in Fig. 67. Sub-pixel translation in a given dimension is defined by:

$$\text{Pixel}_{\text{out}} = \text{Pixel}_{\text{in}} * (1 - \text{Translation}) + \text{Pixel}_{\text{in}-1} * \text{Translation}$$

It can also be represented as a form of interpolation:

$$\text{Pixel}_{\text{out}} = \text{Pixel}_{\text{in}-1} + (\text{Pixel}_{\text{in}} - \text{Pixel}_{\text{in}-1}) * \text{Translation}$$

Implementation of a single (on average) cycle

interpolation engine using a single Multiply ALU and a single Adder ALU in conjunction is straightforward. Sub-pixel translation in both X & Y requires 2 interpolation engines.

In order to sub-pixel translate in Y, 2 Sequential Read Iterators 400, 401 are required (one is reading a line ahead of the other from the same image), and a single Sequential Write Iterator 403 is required.

The first interpolation engine (interpolation in Y) accepts pairs of data from 2 streams, and linearly interpolates between them. The second interpolation engine (interpolation in X) accepts its data as a single 1

dimensional stream and linearly interpolates between values. Both engines interpolate in 1 cycle on average.

Each interpolation engine 405, 406 is capable of performing the sub-pixel translation in 1 cycle per output pixel on average. The overall time is therefore 1 cycle per output pixel, with requirements of 2 Multiply ALUs and 2 Adder ALUs.

The time taken to output 32 pixels from the sub-pixel translate function is on average 320ns (32 cycles). This is enough time for 4 full cache-line accesses to DRAM, so the use of 3 Sequential Iterators is well within timing limits.

The total time taken to sub-pixel translate an image is therefore 1 cycle per pixel of the output image. A typical image to be sub-pixel translated is a tile of size 128 * 128. The output image size is 129 * 129. The process takes $129 * 129 * 10\text{ns} = 166,410\text{ns}$.

The Image Tiler function also makes use of the sub-pixel translation algorithm, but does not require the writing out of the sub-pixel-translated data, but rather processes it further.

Image Tiler

The high level algorithm for tiling an image is carried out in software. Once the placement of the tile has been determined, the appropriate colored tile must be composited. The actual compositing of each tile onto an image is carried out in hardware via the microcoded ALUs. Compositing a tile involves both a texture application and a color application to a background image. In some cases it is desirable to compare the *actual* amount of texture added to the background in relation to the *intended* amount of texture, and use this to scale the color being applied. In these cases the texture must be applied first.

Since color application functionality and texture application functionality are somewhat independent, they are separated into sub-functions.

The number of cycles per 4-channel tile composite for

the different texture styles and coloring styles is summarised in the following table:

	Constant color	Pixel color
Replace texture	4	4.75
25% background + tile texture	4	4.75
Average height algorithm	5	5.75
Average height algorithm with feedback	5.75	6.5

5 Tile Coloring and Compositing

A tile is set to have either a constant color (for the whole tile), or takes each pixel value from an input image. Both of these cases may also have feedback from a texturing stage to scale the opacity (similar to thinning paint).

10 The steps for the 4 cases can be summarised as:

- Sub-pixel translate the tile's opacity values,
- Optionally scale the tile's opacity (if feedback from texture application is enabled).

15 - Determine the color of the pixel (constant or from an image map).

- Composite the pixel onto the background image.

Each of the 4 cases is treated separately, in order to minimize the time taken to perform the function. The summary of time per color compositing style for a single color
20 channel is described in the following table:

Tiling color style	No feedback from texture (cycles per pixel)	Feedback from texture (cycles per pixel)
Tile has constant color per	1	2

pixel		
Tile has per pixel color from input image	1.25	2

Constant color

In this case, the tile has a constant color, determined by software. While the ACP 31 is placing down one tile, the
5 software can be determining the placement and coloring of the next tile.

The color of the tile can be determined by bi-linear interpolation into a scaled version of the image being tiled. The scaled version of the image can be created and
10 stored in place of the image pyramid, and needs only to be performed once per entire tile operation. If the tile size is 128 x 128, then the image can be scaled down by 128:1 in each dimension.

Without feedback

15 When there is no feedback from the texturing of a tile, the tile is simply placed at the specified coordinates. The tile color is used for each pixel's color, and the opacity for the composite comes from the tile's sub-pixel translated opacity channel. In this case color channels and the texture
20 channel can be processed completely independently between tiling passes.

The overview of the process is illustrated in Fig.68. Sub-pixel translation 410 of a tile can be accomplished using 2 Multiply ALUs and 2 Adder ALUs in an average time of
25 1 cycle per output pixel. The output from the sub-pixel translation is the mask to be used in compositing 411 the constant tile color 412 with the background image from background sequential Read Iterator.

Compositing can be performed using 1 Multiply ALU and 1
30 Adder ALU in an average time of 1 cycle per composite. Requirements are therefore 3 Multiply ALUs and 3 Adder ALUs. 4 Sequential Iterators 413-416 are required, taking 320ns to read or write their contents. With an average number of

cycles of 1 per pixel to sub-pixel translate and composite, there is sufficient time to read and write the buffers.

With feedback

When there is feedback from the texturing of a tile, the tile is placed at the specified coordinates. The tile color is used for each pixel's color, and the opacity for the composite comes from the tile's sub-pixel translated opacity channel scaled by the feedback parameter. Thus the texture values must be calculated before the color value is applied.

The overview of the process is illustrated in Fig.62. Sub-pixel translation of a tile can be accomplished using 2 Multiply ALUs and 2 Adder ALUs in an average time of 1 cycle per output pixel. The output from the sub-pixel translation is the mask to be scaled according to the feedback read from the Feedback Sequential Read Iterator 420. The feedback is passed it to a Scaler (1 Multiply ALU) 421.

Compositing 422 can be performed using 1 Multiply ALU and 1 Adder ALU in an average time of 1 cycle per composite. Requirements are therefore 4 Multiply ALUs and all 4 Adder ALUs. Although the entire process can be accomplished in 1 cycle on average, the bottleneck is the memory access, since 5 Sequential Iterators are required. With sufficient buffering, the average time is 1.25 cycles per pixel.

Color from Input Image

One way of coloring pixels in a tile is to take the color from pixels in an input image. Again, there are two possibilities for compositing: with and without feedback from the texturing.

Without feedback

In this case, the tile color simply comes from the relative pixel in the input image. The opacity for compositing comes from the tile's opacity channel sub-pixel shifted.

The overview of the process is illustrated in Fig.70. Sub-pixel translation 425 of a tile can be accomplished

using 2 Multiply ALUs and 2 Adder ALUs in an average time of 1 cycle per output pixel. The output from the sub-pixel translation is the mask to be used in compositing 426 the tile's pixel color (read from the input image 428) with the background image 429.

Compositing 426 can be performed using 1 Multiply ALU and 1 Adder ALU in an average time of 1 cycle per composite. Requirements are therefore 3 Multiply ALUs and 3 Adder ALUs. Although the entire process can be accomplished in 1 cycle on average, the bottleneck is the memory access, since 5 Sequential Iterators are required. With sufficient buffering, the average time is 1.25 cycles per pixel.

With feedback

In this case, the tile color still comes from the relative pixel in the input image, but the opacity for compositing is affected by the relative amount of texture height actually applied during the texturing pass. This process is as illustrated in Fig. 71.

Sub-pixel translation 431 of a tile can be accomplished using 2 Multiply ALUs and 2 Adder ALUs in an average time of 1 cycle per output pixel. The output from the sub-pixel translation is the mask to be scaled 431 according to the feedback read from the Feedback Sequential Read Iterator 432. The feedback is passed to a Scaler (1 Multiply ALU) 431.

Compositing 434 can be performed using 1 Multiply ALU and 1 Adder ALU in an average time of 1 cycle per composite.

Requirements are therefore all 4 Multiply ALUs and 3 Adder ALUs. Although the entire process can be accomplished in 1 cycle on average, the bottleneck is the memory access, since 6 Sequential Iterators are required. With sufficient buffering, the average time is 1.5 cycles per pixel.

Tile Texturing

Each tile has a surface texture defined by its texture channel. The texture must be sub-pixel translated and then applied to the output image. There are 3 styles of texture

compositing:

Replace texture

25% background + tile's texture

Average height algorithm

- 5 In addition, the Average height algorithm can save feedback parameters for color compositing.

The time taken per texture compositing style is summarised in the following table:

Tiling color style	Cycles per pixel (no feedback from texture)	Cycles per pixel (feedback from texture)
Replace texture	1	-
25% background + tile texture value	1	-
Average height algorithm	2	2

10

Replace texture

- In this instance, the texture from the tile replaces the texture channel of the image, as illustrated in Fig.72. Sub-pixel translation 436 of a tile's texture can be accomplished using 2 Multiply ALUs and 2 Adder ALUs in an average time of 1 cycle per output pixel. The output from this sub-pixel translation is fed directly to the Sequential Write Iterator 437.

- 15 The time taken for replace texture compositing is 1 cycle per pixel. There is no feedback, since 100% of the texture value is always applied to the background. There is therefore no requirement for processing the channels in any particular order.

25% Background + Tile's Texture

In this instance, the texture from the tile is added to 25% of the existing texture value. The new value must be greater than or equal to the original value. In addition, the new texture value must be clipped at 255 since the texture channel is only 8 bits. The process utilised is illustrated in Fig.73.

Sub-pixel translation 440 of a tile's texture can be accomplished using 2 Multiply ALUs and 2 Adder ALUs in an average time of 1 cycle per output pixel. The output from this sub-pixel translation 440 is fed to an adder 441 where it is added to $\frac{1}{4}$ 442 of the background texture value. Min and Max functions 444 are provided by the 2 adders not used for sub-pixel translation and the output written to a Sequential Write Iterator 445.

The time taken for this style of texture compositing is 1 cycle per pixel. There is no feedback, since 100% of the texture value is considered to have been applied to the background (even if clipping at 255 occurred). There is therefore no requirement for processing the channels in any particular order.

Average height algorithm

In this texture application algorithm, the average height under the tile is computed, and each pixel's height is compared to the average height. If the pixel's height is less than the average, the stroke height is added to the background height. If the pixel's height is greater than or equal to the average, then the stroke height is added to the average height. Thus background peaks thin the stroke. The height is constrained to increase by a minimum amount to prevent the background from thinning the stroke application to 0 (the minimum amount can be 0 however). The height is also clipped at 255 due to the 8-bit resolution of the texture channel.

There can be feedback of the difference in texture applied versus the expected amount applied. The feedback amount can be used as a scale factor in the application of

the tile's color.

In both cases, the average texture is provided by software, calculated by performing a bi-level interpolation on a scaled version of the texture map. Software determines the next tile's average texture height while the current tile is being applied. Software must also provide the minimum thickness for addition, which is typically constant for the entire tiling process.

Without feedback

With no feedback, the texture is simply applied to the background texture, as shown in Fig.74.

4 Sequential Iterators are required, which means that if the process can be pipelined for 1 cycle, the memory is fast enough to keep up.

Sub-pixel translation 450 of a tile's texture can be accomplished using 2 Multiply ALUs and 2 Adder ALUs in an average time of 1 cycle per output pixel. Each Min & Max function 451,452 requires a separate Adder ALU in order to complete the entire operation in 1 cycle. Since 2 are already used by the sub-pixel translation of the texture, there are not enough remaining for a 1 cycle average time.

The average time for processing 1 pixel's texture is therefore 2 cycles. Note that there is no feedback, and hence the color channel order of compositing is irrelevant.

With feedback

This is conceptually the same as the case without feedback, except that in addition to the standard processing of the texture application algorithm, it is necessary to also record the proportion of the texture actually applied. The proportion can be used as a scale factor for subsequent compositing of the tile's color onto the background image. A flow diagram is illustrated in Fig.75 and the following lookup table is used:

Lookup	Size	Details
LU ₁	256 entries 16 bits per	1/N Table indexed by N (range 0-255)

	entry	Resultant 16 bits treated as fixed point 0:16
--	-------	---

Each of the 256 entries in the software provided 1/N table 460 is 16 bits, thus requiring 16 cache lines to hold continuously.

5 Sub-pixel translation 461 of a tile's texture can be accomplished using 2 Multiply ALUs and 2 Adder ALUs in an average time of 1 cycle per output pixel. Each Min 462 & Max 463 function requires a separate Adder ALU in order to complete the entire operation in 1 cycle. Since 2 are already used by the sub-pixel translation of the texture, 10 there are not enough remaining for a 1 cycle average time.

The average time for processing 1 pixel's texture is therefore 2 cycles. Sufficient space must be allocated for the feedback data area (a tile sized image channel). The texture must be applied before the tile's color is applied, 15 since the feedback is used in scaling the tile's opacity.

CCD Image Interpolator

Images obtained from the CCD via the ISI 83 (Fig.3) are 750 x 500 pixels. When the image is captured via the ISI, the orientation of the camera is used to rotate the pixels 20 by 0, 90, 180, or 270 degrees so that the top of the image corresponds to 'up'. Since every pixel only has an R, G, or B color component (rather than all 3), the fact that these have been rotated must be taken into account when interpreting the pixel values. Depending on the orientation 25 of the camera, each 2x2 pixel block has one of the configurations illustrated in Fig.76:

Several processes need to be performed on the CCD captured image in order to transform it into a useful form for processing:

30 - Up-interpolation of low-sample rate color components in CCD image (interpreting correct orientation of pixels)

Color conversion from RGB to the internal color space

- Scaling of the internal space image from 750 x 500

to 1500 x 1000.

- Writing out the image in a planar format

The entire channel of an image is required to be available at the same time in order to allow warping. In a
5 low memory model (8MB), there is only enough space to hold a single channel at full resolution as a temporary object. Thus the color conversion is to a single color channel. The limiting factor on the process is the color conversion, as it involves tri-linear interpolation from RGB to the
10 internal color space, a process that takes 0.026ns per channel ($750 \times 500 \times 7$ cycles per pixel \times 10ns per cycle = 26,250,000ns).

It is important to perform the color conversion *before* scaling of the internal color space image as this reduces
15 the number of pixels scaled (and hence the overall process time) by a factor of 4.

The requirements for all of the transformations may not fit in the ALU scheme. The transformations are therefore broken into two phases:

20 Phase 1: Up-interpolation of low-sample rate color components in CCD image (interpreting correct orientation of pixels)

Color conversion from RGB to the internal color space
Writing out the image in a planar format

25 Phase 2: Scaling of the internal space image from 750×500 to 1500×1000

Separating out the scale function implies that the small color converted image must be in memory at the same time as the large one. The output from Phase 1 (0.5 MB) can
30 be safely written to the memory area usually kept for the image pyramid (1 MB). The output from Phase 2 can be the general expanded CCD image. Separation of the scaling also allows the scaling to be accomplished by the Affine Transform, and also allows for a different CCD resolution
35 that may not be a simple 1:2 expansion.

Phase 1: Up-interpolation of low-sample rate color

components.

Each of the 3 color components (R, G, and B) needs to be up interpolated in order for color conversion to take place for a given pixel. We have 7 cycles to perform the interpolation per pixel since the color conversion takes 7 cycles.

Interpolation of G is straightforward and is illustrated in Fig.77. Depending on orientation, the actual pixel value G alternates between odd pixels on odd lines & even pixels on even lines, and odd pixels on even lines & even pixels on odd lines. In both cases, linear interpolation is all that is required. Interpolation of R and B components as illustrated in Fig.78 and 79, is more complicated, since in the horizontal and vertical directions, as can be seen from the diagrams, access to 3 rows of pixels simultaneously is required, so 3 Sequential Read Iterators are required, each one offset by a single row. In addition, we have access to the previous pixel on the same row via a latch for each row.

Each pixel therefore contains one component from the CCD, and the other 2 up-interpolated. When one component is being bi-linearly interpolated, the other is being linearly interpolated. Since the interpolation factor is a constant 0.5, interpolation can be calculated by an add and a shift 1 bit right (in 1 cycle), and bi-linear interpolation of factor 0.5 can be calculated by 3 adds and a shift 2 bits right (3 cycles). The total number of cycles required is therefore 4, using a single multiply ALU.

Fig. 80 illustrates the case for rotation 0 even line even pixel (EL, EP), and odd line odd pixel (OL, OP) and Fig. 81 illustrates the case for rotation 0 even line odd pixel (EL, OP), and odd line even pixel (OL, EP). The other rotations are simply different forms of these two expressions.

Color conversion

Color space conversion from RGB to Lab is achieved

using the same method as that described in the general Color Space Convert function, a process that takes 8 cycles per pixel. Phase 1 processing can be described with reference to Fig. 82.

- 5 The up-interpolate of the RGB takes 4 cycles (1 Multiply ALU), but the conversion of the color space takes 8 cycles per pixel (2 Multiply ALUs) due to the lookup transfer time.

Phase 2

10 Scaling the image

- This phase is concerned with up-interpolating the image from the CCD resolution (750 x 500) to the working photo resolution (1500 x 1000). Scaling is accomplished by running the Affine transform with a scale of 1:2. The timing of a
15 general affine transform is 2 cycles per output pixel, which in this case means an elapsed scaling time of 0.03 seconds.

Illuminate Image

Once an image has been processed, it can be illuminated by one or more light sources. Light sources can be:

- 20 1. Directional - is infinitely distant so it casts parallel light in a single direction
 2. Omni - casts unfocused lights in all directions.
 3. Spot - casts a focused beam of light at a specific target point. There is a cone and penumbra associated with a
25 spotlight.

 The scene may also have an associated bump-map to cause reflection angles to vary. Ambient light is also optionally present in an illuminated scene.

- In the process of accelerated illumination, we are
30 concerned with illuminating one image channel by a single light source. Multiple light sources can be applied to a single image channel as multiple passes one pass per light source. Multiple channels can be processed one at a time with or without a bump-map.

- 35 The normal surface vector (N) at a pixel is computed from the bump-map if present. The default normal vector, in

the absence of a bump-map, is perpendicular to the image plane i.e. $N = [0, 0, 1]$.

The viewing vector V is always perpendicular to the image plane i.e. $V = [0, 0, 1]$.

5 For a directional light source, the light source vector (L) from a pixel to the light source is constant across the entire image, so is computed once for the entire image. For an omni light source (at a finite distance), the light source vector is computed independently for each pixel.

10 A pixel's reflection of ambient light is computed according to: $I_a k_a O_d$

A pixel's diffuse and specular reflection of a light source is computed according to the Phong model:

$$f_{att} I_p [k_d O_d (N \cdot L) + k_s O_s (R \cdot V)^n]$$

15 When the light source is at infinity, the light source intensity is constant across the image.

Each light source has three contributions per pixel

20 Ambient Contribution
Diffuse contribution
Specular contribution

The light source can be defined using the following variables:

d_L	Distance from light source
f_{att}	Attenuation with distance [$f_{att} = 1 / d_L^2$]
R	Normalised reflection vector [$R = 2N(N \cdot L) - L$]
I_a	Ambient light intensity
I_p	Diffuse light coefficient
k_a	Ambient reflection coefficient
k_d	Diffuse reflection coefficient
k_s	Specular reflection coefficient
k_{sc}	Specular color coefficient
L	Normalised light source vector
N	Normalised surface normal vector

n	Specular exponent
O _d	Object's diffuse color (i.e. image pixel color)
O _s	Object's specular color ($k_{sc}O_d + (1 - k_{sc})I_p$)
V	Normalised viewing vector [$V = [0, 0, 1]$]

The same reflection coefficients (k_a , k_s , k_d) are used for each color component.

- 5 A given pixel's value will be equal to the ambient contribution plus the sum of each light's diffuse and specular contribution.

Sub-Processes of Illumination Calculation

- 10 In order to calculate diffuse and specular contributions, a variety of other calculations are required. These are calculations of:

- $1/\sqrt{X}$
- N
- L
- $N \cdot L$
- 15 ◦ $R \cdot V$
- f_{att}
- f_{cp}

Sub-processes are also defined for calculating the contributions of:

- 20 ◦ ambient
- diffuse
 - specular

- 25 The sub-processes can then be used to calculate the overall illumination of a light source. Since there are only 4 multiply ALUs, the microcode for a particular type of light source can have sub-processes intermingled appropriately for performance.

Calculation of $1/\sqrt{X}$

- 30 The Vark lighting model uses vectors. In many cases it

is important to calculate the inverse of the length of the vector for normalization purposes. Calculating the inverse of the length requires the calculation of $1/\text{SquareRoot}[X]$.

Logically, the process can be represented as a process with inputs and outputs as shown in Fig.83. Referring to Fig. 84, the calculation can be made via a lookup of the estimation, followed by a single iteration of the following function:

$$V_{n+1} = \frac{1}{2} V_n (3 - X V_n^2)$$

The number of iterations depends on the accuracy required. In this case only 16 bits of precision are required. The table can therefore have 8 bits of precision, and only a single iteration is necessary. The following constant is set by software:

Constant	Value
K_1	3

The following lookup table is used:

Lookup	Size	Details
LU_1	256 entries 8 bits per entry	$1/\text{SquareRoot}[X]$ Table indexed by the 8 highest significant bits of X. Resultant 8 bits treated as fixed point 0:8

Calculation of N

N is the surface normal vector. When there is no bump-map, N is constant. When a bump-map is present, N must be calculated for each pixel.

No bump-map

When there is no bump-map, there is a fixed normal N that has the following properties:

$$N = [X_N, Y_N, Z_N] = [0, 0, 1]$$

$$||N|| = 1$$

$$1/||N|| = 1$$

$$\text{normalized } N = N$$

These properties can be used instead of specifically calculating the normal vector and $1/||N||$ and thus optimize other calculations.

With bump-map

5 As illustrated in Fig. 85, when a bump-map is present, N is calculated by comparing bump-map values in X and Y dimensions. Fig. 85 shows the calculation of N for pixel P1 in terms of the pixels in the same row and column, but not including the value at P1 itself. The calculation of N is
10 made resolution independent by multiplying by a scale factor (same scale factor in X & Y). This process can be represented as a process having inputs and outputs (Z_N is always 1) as illustrated in Fig.86.

 As Z_N is always 1. Consequently X_N and Y_N are not
15 normalized yet (since $Z_N = 1$). Normalization of N is delayed until after calculation of N.L so that there is only 1 multiply by $1/||N||$ instead of 3.

 An actual process for calculating N is illustrated in Fig.87.
20 The following constant is set by software:

Constant	Value
K_1	ScaleFactor (to make N resolution independent)

Calculation of L

Directional lights

25 When a light source is infinitely distant, it has an effective constant light vector L. L is normalized and calculated by software such that:

$L = [X_L, Y_L, Z_L]$
 $||L|| = 1$
30 $1/||L|| = 1$

 These properties can be used instead of specifically calculating the L and $1/||L||$ and thus optimize other calculations. This process is as illustrated in Fig. 88.

Omni lights and Spotlights

When the light source is not infinitely distant, L is the vector from the current point P to the light source PL.

Since $P = [X_P, Y_P, 0]$, L is given by:

5 $L = [X_L, Y_L, Z_L]$
 $X_L = X_P - X_{PL}$
 $Y_L = Y_P - Y_{PL}$
 $Z_L = -Z_{PL}$

We normalize X_L , Y_L and Z_L by multiplying each by $1/||L||$.

10 The calculation of $1/||L||$ (for later use in normalizing) is accomplished by calculating

$$V = X_L^2 + Y_L^2 + Z_L^2$$

and then calculating $V^{-1/2}$

 In this case, the calculation of L can be represented
15 as a process with the inputs and outputs as indicated in Fig. 89.

X_P and Y_P are the coordinates of the pixel whose illumination is being calculated. Z_P is always 0.

 The actual process for calculating L can be as set out
20 in Fig.90.

Where the following constants are set by software:

Constant	Value
K_1	X_{PL}
K_2	Y_{PL}
K_3	Z_{PL}^2 (as Z_P is 0)
K_4	$-Z_{PL}$

Calculation of N.L

 Calculating the dot product of vectors N and L is
25 defined as:

$$X_N X_L + Y_N Y_L + Z_N Z_L$$

No bump-map

 When there is no bump-map N is a constant $[0, 0, 1]$.
N.L therefore reduces to Z_L .

30 With bump-map

When there is a bump-map, we must calculate the dot product directly. Rather than take in normalized N components, we normalize after taking the dot product of a non-normalized N to a normalized L. L is either normalized by software (if it is constant), or by the Calculate L process. This process is as illustrated in Fig. 91.

Note that Z_N is not required as input since it is defined to be 1. However $1/||N||$ is required instead, in order to normalize the result. One actual process for calculating N.L is as illustrated in Fig. 92.

Calculation of R•V

R•V is required as input to specular contribution calculations. Since $V = [0, 0, 1]$, only the Z components are required. R•V therefore reduces to:

$$R \bullet V = 2Z_N(N.L) - Z_L$$

In addition, since the un-normalized $Z_N = 1$, normalized $Z_N = 1/||N||$

No bump-map

The simplest implementation is when N is constant (i.e. no bump-map). Since N and V are constant, N.L and R•V can be simplified:

$$\begin{aligned} V &= [0, 0, 1] \\ N &= [0, 0, 1] \\ L &= [X_L, Y_L, Z_L] \\ N.L &= Z_L \\ R \bullet V &= 2Z_N(N.L) - Z_L \\ &= 2Z_E - Z_L \\ &= Z_L \end{aligned}$$

When L is constant (Directional light source), a normalized Z_L can be supplied by software in the form of a constant whenever R•V is required. When L varies (Omni lights and Spotlights), normalized Z_L must be calculated on the fly. It is obtained as output from the Calculate L process.

With bump-map

When N is not constant, the process of calculating R•V

is simply an implementation of the generalized formula:

$$R \circ V = 2Z_N(N.L) - Z_L$$

The inputs and outputs are as shown in Fig. 93 with the an actual implementation as shown in Fig. 94.

5 Calculation of Attenuation Factor

Directional lights

When a light source is infinitely distant, the intensity of the light does not vary across the image. The attenuation factor f_{att} is therefore 1. This constant can be used to optimize illumination calculations for infinitely distant light sources.

Omni lights and Spotlights

When a light source is not infinitely distant, the intensity of the light can vary according to the following formula:

$$f_{att} = f_0 + f_1/d + f_2/d^2$$

Appropriate settings of coefficients f_0 , f_1 , and f_2 allow light intensity to be attenuated by a constant, linearly with distance, or by the square of the distance.

Since $d = ||L||$, the calculation of f_{att} can be represented as a process with the following inputs and outputs as illustrated in Fig.95.

The actual process for calculating f_{att} can be defined in Fig.96.

Where the following constants are set by software:

Constant	Value
K_1	F_2
K_2	f_1
K_3	F_0

Calculation of Cone and Penumbra Factor

Directional lights and Omni lights

These two light sources are not focused, and therefore have no cone or penumbra. The cone-penumbra scaling factor f_{cp} is therefore 1. This constant can be used to optimize

illumination calculations for Directional and Omni light sources.

Spotlights

A spotlight focuses on a particular target point (PT).
5 The intensity of the Spotlight varies according to whether the particular point of the image is in the cone, in the penumbra, or outside the cone/penumbra region.

Turning now to Fig.97, there is illustrated a graph of f_{cp} with respect to the penumbra position. Inside the cone
10 470, f_{cp} is 1, outside 471 the penumbra f_{cp} is 0. From the edge of the cone through to the end of the penumbra, the light intensity varies according to a cubic function 472.

The various vectors for penumbra 475 and cone 476 calculation are as illustrated in Fig.98 and 99.

15 Looking at the surface of the image in 1 dimension as shown in Fig.99, 3 angles A, B, and C are defined. A is the angle between the target point 479, the light source 478, and the end of the cone 480. C is the angle between the target point 479, light source 478, and the end of the
20 penumbra 481. Both are fixed for a given light source. B is the angle between the target point 479, the light source 478, and the position being calculated 482, and therefore changes with every point being calculated on the image.

We normalize the range A to C to be 0 to 1, and find
25 the distance that B is along that angle range by the formula:

$$(B-A) / (C-A)$$

The range is forced to be in the range 0 to 1 by truncation, and this value used as a lookup for the cubic
30 approximation of f_{cp} .

The calculation of f_{att} can therefore be represented as a process with the inputs and outputs as illustrated in Fig. 100 with an actual process for calculating f_{cp} is as shown in Fig.101 where the following constants are set by software:

35

Constant	Value
----------	-------

K ₁	X _{LT}
K ₂	Y _{LT}
K ₃	Z _{LT}
K ₄	A
K ₅	1/(C-A). [MAXNUM if no penumbra]

The following lookup tables are used:

Lookup	Size	Details
LU ₁	64 entries 16 bits per entry	Arcos(X) Units are same as for constants K ₅ and K ₆ Table indexed by highest 6 bits Result by linear interpolation of 2 entries Timing is 2 * 8 bits * 2 entries = 4 cycles
LU ₂	64 entries 16 bits per entry	Light Response function f _{cp} F(1) = 0, F(0) = 1, others are according to cubic Table indexed by 6 bits (1:5) Result by linear interpolation of 2 entries Timing is 2 * 8 bits = 4 cycles

Calculation of Ambient Contribution

- 5 Regardless of the number of lights being applied to an image, the ambient light contribution is performed once for each pixel, and does not depend on the bump-map.

10 The ambient calculation process can be represented as a process with the inputs and outputs as illustrated in Fig.96. The implementation of the process requires multiplying each pixel from the input image (O_d) by a constant value (I_ak_a), as shown in Fig.103 where the following constant is set by software:

Constant	Value
K_1	$I_a k_a$

Calculation of Diffuse Contribution

Each light that is applied to a surface produces a diffuse illumination. The diffuse illumination is given by the formula:

$$\text{diffuse} = k_d O_d (N.L)$$

There are 2 different implementations to consider:

Implementation 1 - constant N and L

When N and L are both constant (Directional light and no bump-map):

$$N.L = Z_L$$

Therefore:

$$\text{diffuse} = k_d O_d Z_L$$

Since O_d is the only variable, the actual process for calculating the diffuse contribution is as illustrated in Fig. 104 where the following constant is set by software:

Constant	Value
K_1	$k_d (N.L) = k_d Z_L$

Implementation 2 - non-constant N & L

When either N or L are non-constant (either a bump-map or illumination from an Omni light or a Spotlight), the diffuse calculation is performed directly according to the formula:

$$\text{diffuse} = k_d O_d (N.L)$$

The diffuse calculation process can be represented as a process with the inputs as illustrated in Fig. 105. $N.L$ can either be calculated using the Calculate $N.L$ Process, or is provided as a constant. An actual process for calculating the diffuse contribution is as shown in Fig. 106 where the following constants are set by software:

Constant	Value
K_1	k_d

Calculation of Specular Contribution

Each light that is applied to a surface produces a specular illumination. The specular illumination is given by the formula:

$$\text{specular} = k_s O_s (R \cdot V)^n$$

$$\text{where } O_s = k_{sc} O_d + (1 - k_{sc}) I_p$$

There are two implementations of the Calculate Specular process.

10 Implementation 1 - constant N and L

The first implementation is when both N and L are constant (Directional light and no bump-map). Since N, L and V are constant, N.L and R.V are also constant:

$$V = [0, 0, 1]$$

$$15 \quad N = [0, 0, 1]$$

$$L = [X_L, Y_L, Z_L]$$

$$N \cdot L = Z_L$$

$$R \cdot V = 2Z_N(N \cdot L) - Z_L$$

$$= 2Z_L - Z_L$$

$$20 \quad = Z_L$$

The specular calculation can thus be reduced to:

$$\text{specular} = k_s O_s Z_L^n$$

$$= k_s Z_L^n (k_{sc} O_d + (1 - k_{sc}) I_p)$$

$$25 \quad = k_s k_{sc} Z_L^n O_d + (1 - k_{sc}) I_p k_s Z_L^n$$

Since only O_d is a variable in the specular calculation, the calculation of the specular contribution can therefore be represented as a process with the inputs and outputs as indicated in Fig. 107 and an actual process for calculating the specular contribution is illustrated in Fig. 108 where the following constants are set by software:

Constant	Value
K_1	$k_s k_{sc} Z_L^n$
K_2	$(1 - k_{sc}) I_p k_s Z_L^n$

Implementation 2 - non constant N and L

This implementation is when either N or L are not constant (either a bump-map or illumination from an Omni light or a Spotlight). This implies that $R \cdot V$ must be supplied, and hence $R \cdot V^n$ must also be calculated.

- 5 The specular calculation process can be represented as a process with the inputs and outputs as shown in Fig 109. Fig. 110 shows an actual process for calculating the specular contribution where the following constants are set by software:

Constant	Value
K_1	k_s
K_2	k_{sc}
K_3	$(1 - k_{sc}) I_p$

10

The following lookup table is used:

Lookup	Size	Details
LU_1	32 entries 16 bits per entry	X^n Table indexed by 5 highest bits of integer $R \cdot V$ Result by linear interpolation of 2 entries using fraction of $R \cdot V$. Interpolation by 2 Multiplies. The time taken to retrieve the data from the lookup is $2 * 8 \text{ bits} * 2 \text{ entries} = 4$ cycles.

When ambient light is the only illumination

- 15 If the ambient contribution is the only light source, the process is very straightforward since it is not necessary to add the ambient light to anything with the overall process being as illustrated in Fig. 111. We can divide the image vertically into 2 sections, and process each half simultaneously by duplicating the ambient light
- 20 logic (thus using a total of 2 Multiply ALUs and 4 Sequential Iterators). The timing is therefore $\frac{1}{2}$ cycle per pixel for ambient light application.

The typical illumination case is a scene lit by one or more lights. In these cases, because ambient light calculation is so cheap, the ambient calculation is included with the processing of each light source. The first light to be processed should have the correct $I_a k_a$ setting, and subsequent lights should have an $I_a k_a$ value of 0 (to prevent multiple ambient contributions).

If the ambient light is processed as a separate pass (and not the first pass), it is necessary to add the ambient light to the current calculated value (requiring a read and write to the same address). The process overview is shown in Fig.112.

The process uses 3 Image Iterators, 1 Multiply ALU, and takes 1 cycle per pixel on average.

Infinite Light Source

In the case of the infinite light source, we have a constant light source intensity across the image. Thus both L and f_{att} are constant.

No Bump Map

When there is no bump-map, there is a constant normal vector $N [0, 0, 1]$. The complexity of the illumination is greatly reduced by the constants of N , L , and f_{att} . The process of applying a single Directional light with no bump-map is as illustrated in Fig. 112 where the following constant is set by software:

Constant	Value
K_1	I_p

For a single infinite light source we want to perform the logical operations as shown in Fig.113 where K_1 through K_4 are constants with the following values:

Constant	Value
K_1	$K_d (NSL) = K_d L_z$
K_2	k_{sc}
K_3	$K_s (NSH)^n = K_s H_z^2$

K_4	I_p
-------	-------

The process can be simplified since K_2 , K_3 , and K_4 are constants. Since the complexity is essentially in the calculation of the specular and diffuse contributions (using 3 of the Multiply ALUs), it is possible to safely add an ambient calculation as the 4th Multiply ALU. The first infinite light source being processed can have the true ambient light parameter $I_a k_a$, and all subsequent infinite lights can set $I_a k_a$ to be 0. The ambient light calculation becomes effectively free.

If the infinite light source is the first light being applied, there is no need to include the existing contributions made by other light sources and the situation is as illustrated in Fig.114 where the constants have the following values:

Constant	Value
K_1	$k_d(LSN) = k_d L_z$
K_4	I_p
K_5	$(1 - k_s(NSH)^n) I_p = (1 - k_s H_z^n) I_p$
K_6	$k_{sc} k_s (NSH)^n I_p = k_{sc} k_s H_z^n I_p$
K_7	$I_a k_a$

If the infinite light source is not the first light being applied, the existing contribution made by previously processed lights must be included (the same constants apply) and the situation is as illustrated in Fig.113.

In the first case 2 Sequential Iterators 490, 491 are required, and in the second case, 3 Sequential Iterators 490, 491, 492 (the extra Iterator is required to read the previous light contributions). In both cases, the application of an infinite light source with no bump map takes 1 cycle per pixel, including optional application of the ambient light.

With Bump Map

When there is a bump-map, the normal vector N must be calculated per pixel and applied to the constant light source vector L . $1/||N||$ is also used to calculate $R \cdot V$, which is required as input to the Calculate Specular 2 process. The following constants are set by software:

Constant	Value
K_1	X_L
K_2	Y_L
K_3	Z_L
K_4	I_p

Bump-map Sequential Read Iterator 490 is responsible for reading the current line of the bump-map. It provides the input for determining the slope in X . Bump-map Sequential Read Iterators 491, 492 and are responsible for reading the line above and below the current line. They provide the input for determining the slope in Y .

Omni Lights

In the case of the Omni light source, the lighting vector L and attenuation factor f_{att} change for each pixel across an image. Therefore both L and f_{att} must be calculated for each pixel.

No Bump Map

When there is no bump-map, there is a constant normal vector N $[0, 0, 1]$. Although L must be calculated for each pixel, both $N \cdot L$ and $R \cdot V$ are simplified to Z_L . When there is no bump-map, the application of an Omni light can be calculated as shown in Fig. 114 where the following constants are set by software:

Constant	Value
K_1	X_p
K_2	Y_p
K_3	I_p

The algorithm optionally includes the contributions from previous light sources, and also includes an ambient light calculation. Ambient light needs only to be included once. For all other light passes, the appropriate constant
5 in the Calculate Ambient process should be set to 0.

The algorithm as shown requires a total of 19 multiply/accumulates. The times taken for the lookups are 1 cycle during the calculation of L, and 4 cycles during the specular contribution. The processing time of 5 cycles is
10 therefore the best that can be accomplished. The time taken is increased to 6 cycles in case it is not possible to optimally microcode the ALUs for the function. The speed for applying an Omni light onto an image with no associated bump-map is 6 cycles per pixel.

15 With Bump-map

When an Omni light is applied to an image with an associated a bump-map, calculation of N, L, N.L and R.V are all necessary. The process of applying an Omni light onto an image with an associated bump-map is as indicated in Fig.
20 115 where the following constants are set by software:

Constant	Value
K ₁	X _P
K ₂	Y _P
K ₃	I _P

The algorithm optionally includes the contributions from previous light sources, and also includes an ambient light calculation. Ambient light needs only to be included
25 once. For all other light passes, the appropriate constant in the Calculate Ambient process should be set to 0.

The algorithm as shown requires a total of 32 multiply/accumulates. The times taken for the lookups are 1 cycle each during the calculation of both L and N, and 4
30 cycles for the specular contribution. However the lookup required for N and L are both the same (thus 2 LUs implement the 3 LUs). The processing time of 8 cycles is adequate. The

time taken is extended to 9 cycles in case it is not possible to optimally microcode the ALUs for the function. The speed for applying an Omni light onto an image with an associated bump-map is 9 cycles per pixel.

5

Spotlights

Spotlights are similar to Omni lights except that the attenuation factor f_{att} is modified by a cone/penumbra factor f_{cp} that effectively focuses the light around a target.

10 No bump-map

When there is no bump-map, there is a constant normal vector N $[0, 0, 1]$. Although L must be calculated for each pixel, both $N \cdot L$ and $R \cdot V$ are simplified to Z_L . Fig. 116 illustrates the application of a Spotlight to an image where the following constants are set by software:

15

Constant	Value
K_1	X_P
K_2	Y_P
K_3	I_P

The algorithm optionally includes the contributions from previous light sources, and also includes an ambient light calculation. Ambient light needs only to be included once. For all other light passes, the appropriate constant in the Calculate Ambient process should be set to 0.

20

The algorithm as shown requires a total of 30 multiply/accumulates. The times taken for the lookups are 1 cycle during the calculation of L , 4 cycles for the specular contribution, and 2 sets of 4 cycle lookups in the cone/penumbra calculation.

25

With bump-map

When a Spotlight is applied to an image with an associated a bump-map, calculation of N , L , $N \cdot L$ and $R \cdot V$ are all necessary. The process of applying a single Spotlight onto an image with associated bump-map is illustrated in Fig. 117 where the following constants are set by software:

30

The algorithm optionally includes the contributions from previous light sources, and also includes an ambient light calculation. Ambient light needs only to be included once. For all other light passes, the appropriate constant in the Calculate Ambient process should be set to 0. The algorithm as shown requires a total of 41 multiply/accumulates.

Print Roll Authentication

Print Head 44

- 10 Fig. 118 illustrates the logical layout of a single print Head which logically consists of 8 segments, each printing bi-level cyan, magenta, and yellow onto a portion of the page.

Loading a segment for printing

- 15 Before anything can be printed, each of the 8 segments in the Print Head must be loaded with 6 rows of data corresponding to the following relative rows in the final output image:

- 20 Row 0 = Line N, Yellow, even dots 0, 2, 4, 6, 8, ...
Row 1 = Line N+8, Yellow, odd dots 1, 3, 5, 7, ...
Row 2 = Line N+10, Magenta, even dots 0, 2, 4, 6, 8, ...
Row 3 = Line N+18, Magenta, odd dots 1, 3, 5, 7, ...
Row 4 = Line N+20, Cyan, even dots 0, 2, 4, 6, 8, ...
Row 5 = Line N+28, Cyan, odd dots 1, 3, 5, 7, ...

- 25 Each of the segments prints dots over different parts of the page. Each segment prints 750 dots of one color, 375 even dots on one row, and 375 odd dots on another. The 8 segments have dots corresponding to positions:

Segment	First dot	Last dot
0	0	749
1	750	1499
2	1500	2249
3	2250	2999
4	3000	3749
5	3750	4499

6	4500	5249
7	5250	5999

Each dot is represented in the Print Head segment by a single bit. The data must be loaded 1 bit at a time by placing the data on the segment's BitValue pin, and clocked
5 in to a shift register in the segment according to a BitClock. Since the data is loaded into a shift register, the order of loading bits must be correct. Data can be clocked in to the Print Head at a maximum rate of 10 MHz.

Once all the bits have been loaded, they must be
10 transferred in parallel to the Print Head output buffer, ready for printing. The transfer is accomplished by a single pulse on the segment's ParallelXferClock pin.

Controlling the Print

In order to conserve power, not all the dots of the
15 Print Head have to be printed simultaneously. A set of control lines enables the printing of specific dots. An external controller, such as the ACP, can change the number of dots printed at once, as well as the duration of the print pulse in accordance with speed and/or power
20 requirements.

Each segment has 5 NozzleSelect lines, which are decoded to select 32 sets of nozzles per row. Since each row has 375 nozzles, each set contains 12 nozzles. There are also 2 BankEnable lines, one for each of the odd and even
25 rows of color. Finally, each segment has 3 ColorEnable lines, one for each of C, M, and Y colors. A pulse on one of the ColorEnable lines causes the specified nozzles of the color's specified rows to be printed. A pulse is typically about 2μs in duration.

30 If all the segments are controlled by the same set of NozzleSelect, BankEnable and ColorEnable lines (wired externally to the print head), the following is true:

If both odd and even banks print simultaneously (both BankEnable bits are set), 24 nozzles fire simultaneously per

segment, 192 nozzles in all, consuming 5.7 Watts.

If odd and even banks print independently, only 12 nozzles fire simultaneously per segment, 96 in all, consuming 2.85 Watts.

5 Print Head Interface 62

The Print Head Interface 62 connects the ACP to the Print Head, providing both data and appropriate signals to the external Print Head. The Print Head Interface 62 works in conjunction with both a VLIW processor 74 and a software
10 algorithm running on the CPU in order to print a photo in approximately 2 seconds.

An overview of the inputs and outputs to the Print Head Interface is shown in Fig. 119. The Address and Data Buses are used by the CPU to address the various registers in the
15 Print Head Interface. A single BitClock output line connects to all 8 segments on the print head. The 8 DataBits lines lead one to each segment, and are clocked in to the 8 segments on the print head simultaneously (on a BitClock pulse). For example, dot 0 is transferred to segment₀, dot
20 750 is transferred to segment₁, dot 1500 to segment₂, etc. simultaneously.

The VLIW Output FIFO contains the dithered bi-level C, M, and Y 6000 x 9000 resolution print image in the correct order for output to the 8 DataBits. The ParallelXferClock
25 is connected to each of the 8 segments on the print head, so that on a single pulse, all segments transfer their bits at the same time. Finally, the NozzleSelect, BankEnable and ColorEnable lines are connected to each of the 8 segments, allowing the Print Head Interface to control the duration of
30 the C, M, and Y drop pulses as well as how many drops are printed with each pulse. Registers in the Print Head Interface allow the specification of pulse durations between 0 and 6 μ s, with a typical duration of 2 μ s.

Printing an Image

35 There are 2 phases that must occur before an image is in the hand of the Artcam user:

1. Preparation of the image to be printed
2. Printing the prepared image

Preparation of an image only needs to be performed once.
Printing the image can be performed as many times as
5 desired.

Prepare the Image

Preparing an image for printing involves:

1. Convert the Photo Image into a Print Image
2. Rotation of the Print Image (internal color space)
10 to align the output for the orientation of the printer
3. Up-interpolation of compressed channels (if
necessary)
4. Color conversion from the internal color space to
the CMY color space appropriate to the specific printer and
15 ink

At the end of image preparation, a 4.5MB correctly
oriented 1000 x 1500 CMY image is ready to be printed.

Convert Photo Image to Print Image

The conversion of a Photo Image into a Print Image
20 requires the execution of a Vark script to perform image
processing. The script is either a default image enhancement
script or a Vark script taken from the currently inserted
Artcard. The Vark script is executed via the CPU,
accelerated by functions performed by the VLIW Vector
25 Processor.

Rotate the Print Image

The image in memory is originally oriented to be top
upwards. This allows for straightforward Vark processing.
Before the image is printed, it must be aligned with the
30 print roll's orientation. The re-alignment only needs to be
done once. Subsequent Prints of a Print Image will already
have been rotated appropriately.

The transformation to be applied is simply the inverse
of that applied during capture from the CCD when the user
35 pressed the "Image Capture" button on the Artcam. If the
original rotation was 0, then no transformation needs to

take place. If the original rotation was +90 degrees, then the rotation before printing needs to be -90 degrees (same as 270 degrees). The method used to apply the rotation is the Vark accelerated Affine Transform function. The Affine Transform engine can be called to rotate each color channel independently. Note that the color channels cannot be rotated in place. Instead, they can make use of the space previously used for the expanded single channel (1.5MB).

Fig. 120 shows an example of rotation of a Lab image where the a and b channels are compressed 4:1. The L channel is rotated into the space no longer required (the single channel area), then the a channel can be rotated into the space left vacant by L, and finally the b channel can be rotated. The total time to rotate the 3 channels is 0.09 seconds. It is an acceptable period of time to elapse before the first print image. Subsequent prints do not incur this overhead.

Up Interpolate and color convert

The Lab image must be converted to CMY before printing. Different processing occurs depending on whether the a and b channels of the Lab image is compressed. If the Lab image is compressed, the a and b channels must be decompressed before the color conversion occurs. If the Lab image is not compressed, the color conversion is the only necessary step. The Lab image must be up interpolated (if the a and b channels are compressed) and converted into a CMY image. A single VLIW process combining scale and color transform can be used.

The method used to perform the color conversion is the Vark accelerated Color Convert function. The Affine Transform engine can be called to rotate each color channel independently. The color channels cannot be rotated in place. Instead, they can make use of the space previously used for the expanded single channel (1.5MB).

Print the Image

Printing an image is concerned with taking a correctly

oriented 1000 x 1500 CMY image, and generating data and signals to be sent to the external Print Head. The process involves the CPU working in conjunction with a VLIW process and the Print Head Interface.

5 The resolution of the image in the Artcam is 1000 x 1500. The printed image has a resolution of 6000 x 9000 dots, which makes for a very straightforward relationship: 1 pixel = $6 \times 6 = 36$ dots. As shown in Fig. 121 since each dot is $16.6\mu\text{m}$, the 6×6 dot square is $100\mu\text{m}$ square. Since
10 each of the dots is bi-level, the output must be dithered.

 The image should be printed in approximately 2 seconds. For 9000 rows of dots this implies a time of $222\mu\text{s}$ time between printing each row. The Print Head Interface must generate the 6000 dots in this time, an average of 37ns per
15 dot. However, each dot comprises 3 colors, so the Print Head Interface must generate each color component in approximately 12ns, or 1 clock cycle of the ACP (10ns at 100 MHz). One VLIW process is responsible for calculating the next line of 6000 dots to be printed. The odd and even C,
20 M, and Y dots are generated by dithering input from 6 different 1000 x 1500 CMY image lines. The second VLIW process is responsible for taking the previously calculated line of 6000 dots, and correctly generating the 8 bits of data for the 8 segments to be transferred by the Print Head
25 Interface to the Print Head in a single transfer.

 A CPU process updates registers in the first VLIW process 3 times per print line (once per color component = 27000 times in 2 seconds), and in the 2nd VLIW process once every print line (9000 times in 2 seconds). The CPU works
30 one line ahead of the VLIW process in order to do this.

 Finally, the Print Head Interface takes the 8 bit data from the VLIW Output FIFO, and outputs it unchanged to the Print Head, producing the BitClock signals appropriately. Once all the data has been transferred a ParallelXferClock
35 signal is generated to load the data for the next print line. In conjunction with transferring the data to the

Print Head, a separate timer is generating the signals for the different print cycles of the Print Head using the NozzleSelect, ColorEnable, and BankEnable lines as specified by Print Head Interface internal registers.

- 5 The CPU also controls the various motors and guillotine via the parallel interface during the print process.

Generate C, M, and Y Dots

- 10 The input to this process is a 1000 x 1500 CMY image correctly oriented for printing. The image is not compressed in any way. As illustrated in Fig. 122, a VLIW microcode program takes the CMY image, and generates the C, M, and Y pixels required by the Print Head Interface to be dithered.

- 15 The process is run 3 times, once for each of the 3 color components. The process consists of 2 sub-processes run in parallel - one for producing even dots, and the other for producing odd dots. Each sub-process takes one pixel from the input image, and produces 3 output dots (since one pixel = 6 output dots, and each sub-process is concerned with either even or odd dots). Thus one output dot is
20 generated each cycle, but an input pixel is only read once every 3 cycles.

- 25 The original dither cell is a 64 x 64 cell, with each entry 8 bits. This original cell is divided into an odd cell and an even cell, so that each is still 64 high, but only 32 entries wide. The even dither cell contains original dither cell pixels 0, 2, 4 etc., while the odd contains original
30 dither cell pixels 1, 3, 5 etc. Since a dither cell repeats across a line, a single 32 byte line of each of the 2 dither cells is required during an entire line, and can therefore be completely cached. The odd and even lines of a single
35 process line are staggered 8 dot lines apart, so it is convenient to rotate the odd dither cell's lines by 8 lines. Therefore the same offset into both odd and even dither cells can be used. Consequently the even dither cell's line corresponds to the even entries of line L in the original
dither cell, and the even dither cell's line corresponds to

the odd entries of line $L+8$ in the original dither cell.

The process is run 3 times, once for each of the color components. The CPU software routine must ensure that the Sequential Read Iterators for odd and even lines are

5 pointing to the correct image lines corresponding to the print heads. For example, to produce one set of 18,000 dots (3 sets of 6000 dots):

- Yellow even dot line = 0, therefore input Yellow image line = $0/6 = 0$

10 ◦ Yellow odd dot line = 8, therefore input Yellow image line = $8/6 = 1$

- Magenta even line = 10, therefore input Magenta image line = $10/6 = 1$

15 ◦ Magenta odd line = 18, therefore input Magenta image line = $18/6 = 3$

- Cyan even line = 20, therefore input Cyan image line = $20/6 = 3$

- Cyan odd line = 28, therefore input Cyan image line = $28/6 = 4$

20 Subsequent sets of input image lines are:

- $Y=[0, 1], M=[1, 3], C=[3, 4]$

- $Y=[0, 1], M=[1, 3], C=[3, 4]$

- $Y=[0, 1], M=[2, 3], C=[3, 5]$

- $Y=[0, 1], M=[2, 3], C=[3, 5]$

25 ◦ $Y=[0, 2], M=[2, 3], C=[4, 5]$

The dither cell data however, does not need to be updated for each color component. The dither cell for the 3 colors becomes the same, but offset by 2 dot lines for each component.

30 The Dithered Output is written to a Sequential Write Iterator, with odd and even dithered dots written to 2 separate outputs. The same two Write Iterators are used for all 3 color components, so that they are contiguous within the break-up of odd and even dots.

While one set of dots is being generated for a print line, the previously generated set of dots is being merged by a second VLIW process as described in the next section.

Generate Merged 8 bit Dot Output

5 This process, as illustrated in Fig. 123, takes a single line of dithered dots and generates the 8 bit data stream for output to the Print Head Interface via the VLIW Output FIFO. The process requires the entire line to have been prepared, since it requires semi-random access to most
10 of the dithered line at once. The following constant is set by software:

Constant	Value
K ₁	375

The Sequential Read Iterators point to the line of previously generated dots, with the Iterator registers set
15 up to limit access to a single color component. The distance between subsequent pixels is 375, and the distance between one line and the next is given to be 1 byte. Consequently 8 entries are read for each "line". A single "line" corresponds to the 8 bits to be loaded on the print head.
20 The total number of "lines" in the image is set to be 375. With at least 8 cache lines assigned to the Sequential Read Iterator, complete cache coherence is maintained. Instead of counting the 8 bits, 8 Microcode steps count implicitly.

The generation process first reads all the entries from
25 the even dots, combining 8 entries into a single byte which is then output to the VLIW Output FIFO. Once all 3000 even dots have been read, the 3000 odd dots are read and processed. A software routine must update the address of the dots in the odd and even Sequential Read Iterators once
30 per color component, which equates to 3 times per line. The two VLIW processes require all 8 ALUs and the VLIW Output FIFO. As long as the CPU is able to update the registers as described in the two processes, the VLIW processor can generate the dithered image dots fast enough to keep up with

the printer.

Data Card Reader

Fig. 124, there is illustrated on form of card reader 500 which allows for the insertion of Artcards 9 for reading. Fig. 123 shows an exploded perspective of the reader of Fig. 124. Cardreader is interconnected to a computer system and includes a CCD reading mechanism 35. The cardreader includes pinch rollers 506, 507 for pinching an inserted Artcard 9. One of the roller e.g. 506 is driven by an Artcard motor 37 for the advancement of the card 9 between the two rollers 506 and 507 at a uniformed speed. The Artcard 9 is passed over a series of LED lights 512 which are encased within a clear plastic mould 514 having a semi circular cross section. The cross section focuses the light from the LEDs eg 512 onto the surface of the card 9 as it passes by the LEDs 512. From the surface it is reflected to a high resolution linear CCD 34 which is constructed to a resolution of approximately 480 dpi. The surface of the Artcard 9 is encoded to the level of approximately 1600 dpi hence, the linear CCD 34 supersamples the Artcard surface with an approximately three times multiplier. The Artcard 9 is further driven at a speed such that the linear CCD 34 is able to supersample in the direction of Artcard movement at a rate of approximately 4800 readings per inch. The scanned Artcard CCD data is forwarded from the Artcard reader to ACP 31 for processing. A sensor 49, which can comprise a light sensor acts to detect of the presence of the card 13.

The CCD reader includes a bottom substrate 516, a top substrate 514 which comprises a transparent molded plastic. In between the two substrates is inserted the linear CCD array 34 which comprises a thin long linear CCD array constructed by means of semi-conductor manufacturing processes.

Turning to Fig. 125, there is illustrated a side perspective view, partly in section, of an example construction of the CCD reader unit. The series of LEDs eg.

512 are operated to emit light when a card 9 is passing across the surface of the CCD reader 34. The emitted light is transmitted through a portion of the top substrate 523. The substrate includes a portion eg. 529 having a curved circumference so as to focus light emitted from LED 512 to a point eg. 532 on the surface of the card 9. The focused light is reflected from the point 532 towards the CCD array 34. A series of microlenses eg. 534, shown in exaggerated form, are formed on the surface of the top substrate 523. The microlenses 523 act to focus light received across the surface to the focused down to a point 536 which corresponds to point on the surface of the CCD reader 34 for sensing of light falling on the light sensing portion of the CCD array 34.

A number of refinements of the above arrangement are possible. For example, the sensing devices on the linear CCD 34 may be staggered. The corresponding microlenses 34 can also be correspondingly formed as to focus light into a staggered series of spots so as to correspond to the staggered CCD sensors.

To assist reading, the data surface area of the Artcard 9 is modulated with a checkerboard pattern as previously discussed with reference to Fig. 37. Other forms of high frequency modulation may be possible however.

It will be evident that an Artcard printer can be provided as for the printing out of data on storage Artcard. Hence, the Artcard system can be utilized as a general form of information distribution outside of the Artcam device. An Artcard printer can prints out Artcards on high quality print surfaces and multiple Artcards can be printed on same sheets and later separated. On a second surface of the Artcard 9 can be printed information relating to the files etc. stored on the Artcard 9 for subsequent storage.

Hence, the Artcard system allows for a simplified form of storage which is suitable for use in place of other forms of storage such as CD ROMs, magnetic disks etc. The

Artcards 9 can also be mass produced and thereby produced in a substantially inexpensive form for redistribution.

Print Rolls

5 Turning to Fig. 127, there is illustrated the print roll 42 and print-head portions of the Artcam. The paper/film 611 is fed in a continuous "web-like" process to a printing mechanism 15 which includes further pinch rollers 616 - 619 and a print head 44

10 The pinch roller 613 is connected to a drive mechanism (not shown) and upon rotation of the print roller 613, "paper" in the form of film 611 is forced through the printing mechanism 615 and out of the picture output slot 6. A rotary guillotine mechanism (not shown) is utilised to cut the roll of paper 611 at required photo
15 sizes.

It is therefore evident that the printer roll 42 is responsible for supplying "paper" 611 to the print mechanism 615 for printing of photographically imaged pictures.

20 In Fig. 128, there is shown an exploded perspective of the print roll 42. The printer roll 42 includes output printer paper 611 which is output under the operation of pinching rollers 612, 613.

Referring now to Fig. 129, there is illustrated a
25 more fully exploded perspective view, of the print roll 42 of Fig. 128 without the "paper" film roll. The print roll 42 includes three main parts comprising ink reservoir section 620, paper roll sections 622, 623 and outer casing sections 626, 627.

30 Turning first to the ink reservoir section 620, which includes the ink reservoir or ink supply sections 633. The ink for printing is contained within three bladder type containers 630 - 632. The printer roll 42 is assumed to provide full color output inks. Hence, a first ink
35 reservoir or bladder container 630 contains cyan colored ink. A second reservoir 631 contains magenta colored ink

and a third reservoir 632 contains yellow ink. Each of the reservoirs 630 - 632, although having different volumetric dimensions, are designed to have substantially the same volumetric size.

5 The ink reservoir sections 621, 633, in addition to cover 624 can be made of plastic sections and are designed to be mated together by means of heat sealing, ultra violet radiation, etc. Each of the equally sized ink reservoirs 630 - 632 is connected to a corresponding ink
10 channel 639 - 641 for allowing the flow of ink from the reservoir 630 - 632 to a corresponding ink output port 635 - 637. The ink reservoir 632 having ink channel 641, and output port 637, the ink reservoir 631 having ink channel 640 and output port 636, and the ink reservoir 630 having
15 ink channel 639 and output port 637.

 In operation, the ink reservoirs 630 - 632 can be filled with corresponding ink and the section 633 joined to the section 621. The ink reservoir sections 630 - 632, being collapsible bladders, allow for ink to traverse ink
20 channels 639 - 641 and therefore be in fluid communication with the ink output ports 635 - 637. Further, if required, an air inlet port can also be provided to allow the pressure associated with ink channel reservoirs 630 - 632 to be maintained as required.

25 The cap 624 can be joined to the ink reservoir section 620 so as to form a pressurized cavity, accessible by the air pressure inlet port.

 The ink reservoir sections 621, 633 and 624 are designed to be connected together as an integral unit and to be inserted inside printer roll sections 622, 623. The
30 printer roll sections 622, 623 are designed to mate together by means of a snap fit by means of male portions 645 - 647 mating with corresponding female portions (not shown). Similarly, female portions 654 - 656 are designed
35 to mate with corresponding male portions 660 - 662. The paper roll sections 622, 623 are therefore designed to be

snapped together. One end of the film within the role is pinched between the two sections 622, 623 when they are joined together. The print film can then be rolled on the print roll sections 622, 625 as required.

5 As noted previously, the ink reservoir sections 620, 621, 633, 624 are designed to be inserted inside the paper roll sections 622, 623. The printer roll sections 622, 623 are able to be rotatable around stationery ink reservoir sections 621, 633 and 624 to dispense film on
10 demand.

 The outer casing sections 626 and 627 are further designed to be coupled around the print roller sections 622, 623. In addition to each end of pinch rollers eg 612, 613 is designed to clip in to a corresponding cavity
15 eg 670 in cover 626, 627 with roller 613 being driven externally (not shown) to feed the print film and out of the print roll.

 Finally, a cavity 677 can be provided in the ink reservoir sections 620, 621 for the insertion and gluing
20 of an silicon chip integrated circuit type device 53 for the storage of information associated with the print roll 42:

 As shown in Fig. 120 and Fig. 129, the print roll 42 is designed to be inserted into the Artcam camera device
25 so as to couple with a coupling unit 680 which includes connector pads 681 for providing a connection with the silicon chip 53. Further, the connector 680 includes end connectors of four connecting with ink supply ports 635 - 637. The ink supply ports are in turn to connect to ink
30 supply lines eg 682 which are in turn interconnected to printheads supply ports eg. 687 for the flow of ink to print-head 44 in accordance with requirements.

 The "media" 611 utilised to form the roll can comprise many different materials on which it is designed
35 to print suitable images. For example, opaque rollable plastic material may be utilized, transparencies may be

used by using transparent plastic sheets, metallic printing can take place via utilization of a metallic sheet film. Further, fabrics could be utilised within the printer roll 42 for printing images on fabric, although
5 care must be taken that only fabrics having a suitable stiffness or suitable backing material are utilised.

When the print media is plastic, it can be coated with a layer which fixes and absorbs the ink. Further, several types of print media may be used, for example,
10 opaque white matte, opaque white gloss, transparent film, frosted transparent film, lenticular array film for stereoscopic 3D prints, metallised film, film with the embossed optical variable devices such as gratings or holograms, media which is pre-printed on the reverse side,
15 and media which includes a magnetic recording layer. When utilising a metallic foil, the metallic foil can have a polymer base, coated with a thin (several micron) evaporated layer of aluminum or other metal and then coated with a clear protective layer adapted to receive
20 the ink via the ink printer mechanism.

In use the print roll 42 is obviously designed to be inserted inside a camera device so as to provide ink and paper for the printing of images on demand. The ink output ports 635 - 637 meet with corresponding ports
25 within the camera device and the pinch rollers 672, 673 are operated to allow the supply of paper to the camera device under the control of the camera device.

As illustrated in Fig. 129, a mounted silicon chip 53 is insert in one end of the print roll 42. In Fig. 130 the
30 authentication chip 53 is shown in more detail and includes four communications leads 680 - 683 for communicating details from the chip 53 to the corresponding camera to which it is inserted.

Turning to Fig. 130, the chip can be separately created
35 by means of encasing a small integrated circuit 687 in epoxy and running bonding leads eg. 688 to the external

communications leads 680 - 683. The integrated chip 687 being approximately 400 microns square with a 100 micron scribe boundary. Subsequently, the chip can be glued to an appropriate surface of the cavity of the print roll 42. In
5 Fig. 131, there is illustrated the integrated circuit 687 interconnected to bonding pads 681, 682 in an exploded view of the arrangement of Fig. 130.

Referring now to Fig. 132, there is illustrated generally 700 the internal architecture of the chip of Fig.
10 131 The chip architecture 700 includes a flash memory store 701, a roll authentication unit 702, a command decoder 703 and a communications controller 704.

The communications controller 704 is interconnected to the serial input and output wires 681, 682 for communication
15 with the Artcam. The command decoder 703 receives commands from the camera Artcam via the communications controller 704 controls the flash memory store 701 and roll authentication unit 702 to carry out the command. Preferably, the flash memory store 701 provides 1,024 bits of information which
20 includes fixed data written into the flash memory at manufacturing time in addition to variable data storage.

Turning now to Fig. 133, there is illustrated 705 the information stored within the flash memory store 701. This data can include the following:

25 Factory Code

The factory code is a 16 bit code indicating the factory at which the print roll was manufactured. This identifies factories belonging to the owner of the print roll technology, or factories making print rolls under
30 license. The purpose of this number is to allow the tracking of factory that a print roll came from, in case there are quality problems.

Batch Number

The batch number is a 32 bit number indicating the
35 manufacturing batch of the print roll. The purpose of this number is to track the batch that a print roll came from, in

case there are quality problems.

Serial Number

A 48 bit serial number is provided to allow unique identification of each print roll up to a maximum of 280
5 trillion print rolls.

Manufacturing date

A 16 bit manufacturing date is included for tracking the age of print rolls, in case the shelf life is limited.

Media length

10 The length of print media remaining on the roll is represented by this number. This length is represented in small units such as millimeters or the smallest dot pitch of printer devices using the print roll and to allow the calculation of the number of remaining photos in each of the
15 well known C, H, and P formats, as well as other formats which may be printed. The use of small units also ensures a high resolution can be used to maintain synchronization with pre-printed media.

Media Type

20 The media type datum enumerates the media contained in the print roll.

- (1) Transparent
- (2) Opaque white
- (3) Opaque tinted
- 25 (4) 3D lenticular
- (5) Pre-printed: length specific
- (6) Pre-printed: not length specific
- (7) Metallic foil
- (8) Holographic/optically variable device foil

30 Pre-printed Media Length

The length of the repeat pattern of any pre-printed media contained, for example on the back surface of the print roll is stored here.

Ink Viscosity

35 The viscosity of each ink color is included as an 8 bit number. the ink viscosity numbers can be used to adjust the

print head actuator characteristics to compensate for viscosity (typically, a higher viscosity will require a longer actuator pulse to achieve the same drop volume).

Recommended Drop Volume for 1200 dpi

- 5 The recommended drop volume of each ink color is included as an 8 bit number. The most appropriate drop volume will be dependent upon the ink and print media characteristics. For example, the required drop volume will decrease with increasing dye concentration or absorptivity.
- 10 Also, transparent media require around twice the drop volume as opaque white media, as light only passes through the dye layer once for transparent media.

- As the print roll contains both ink and media, a custom match can be obtained. The drop volume is only the
- 15 recommended drop volume, as the printer may be other than 1200 dpi, or the printer may be adjusted for lighter or darker printing.

Ink Color

- The color of each of the dye colors is included and can
- 20 be used to "fine tune" the digital half toning that is applied to any image before printing.

Remaining Media Length Indicator

- The length of print media remaining on the roll is represented by this number and is updatable by the camera
- 25 device. The length is represented in small units (eg. 1200 dpi pixels) to allow calculation of the number of remaining photos in each of C, H, and P formats, as well as other formats which may be printed. The high resolution can also be used to maintain synchronization with pre-printed media.

30 Copyright or Bit Pattern

 This 512 bit pattern represents an ASCII character sequence sufficient to allow the contents of the flash memory store to be copyrightable.

Authentication Key

- 35 This key includes authentication data to make it difficult for third parties to reverse engineer the print

roll technology.

Finally, a further 88 bits are reserved for future camera use.

5 The roll authentication unit 702 as will become more apparent hereinafter, takes the authentication key from flash memory store 701 and combines it with a print roll test code received from the ACP.

Authentication

10 The print roll manufacturing process includes in-built measures to stop illegal clone manufacture in countries with weak industrial property protection from copying the technology.

15 The print rolls 42 are not protected against cloning by high technology barriers, such as the extraordinarily difficult chemistry of color silver halide film in photographic reproduction. The present embodiment is simply constructed from plastic injection moulding, coated paper, and ink. the coated paper and ink may only be required to be compatible, and do not need to match some special
20 formulation. To protect against these problems, an authentication code and circuit is included in the print roll chip.

The authentication system prevents illegal copying which can have the disastrous consequence of ink nozzles becoming clogged by poorly filtered ink in "clone" print
25 rolls. This will assist in stopping a consumer blaming the camera manufacturer and in stopping the spread of counterfeit print rolls.

The authentication system should remain outside most countries' legislation in respect of the export of
30 cryptographic materials and should deal with the following issues:.

(1) Reverse Engineering of the Print Roll Chip

35 The best way to protect against reverse engineering of the chip is to make the benefit of reverse engineering minimal. To achieve this, the authentication keys are

stored in non-volatile flash memory store not in ROM.

(2) Brute force cryptanalysis

Brute force cryptanalysis can be prevented by making the authentication key long enough. To be secure against
5 computational improvements over the next fifty years, a long key is necessary. A key length of 128 bits means that 2^{128} tests (3.4×10^{38} tests) must be made to launch a brute force attack. This would take ten billion years on an array or a trillion processors each running 1 billion tests per second.

10 (3) Substitution with a complete lookup table

If the number of test codes sent by the camera to the print roll is small, then there is no need for a clone manufacturer to crack the authentication code. Instead, the clone manufacturer could incorporate a ROM in their print
15 roll which had a record of all of the responses from a genuine print roll to the codes sent by the camera. In 30 years, it may be cost effective to build a terabyte ROM into each clone print roll. Therefore, the camera should send random authentication test words that are at least 40 bits
20 long. A 128 bit authentication test word will certainly be more than adequate.

(4) Substitution with a sparse lookup table

If the test codes sent by the camera are somehow predictable, rather than effectively random, then the clone
25 manufacturer need not provide a complete lookup table. For example:

(a) If the test code is simply the serial number of the camera, the clone manufacturer need simply provide a lookup table which contains values for past and predicted
30 future serial camera serial numbers. There are unlikely to be more than 109 of these.

(b) If the test code is simply the date, then the clone manufacturer can produce a lookup table using the date as the address.

35 (c) If the test code is a pseudo-random number using either the serial number or the date as a seed, then the

clone manufacturer just needs to crack the pseudo-random number generator in the camera. This is probably not difficult, as the clone manufacturer may gain access to the object code of the camera. The clone manufacturer could
5 then produce an content addressable memory (or other sparse array lookup) using these codes to access stored authentication codes.

Therefore, long random test keys should be generated by a relatively secure process. This random number generator
10 can be in the machine which writes the authentication code to the chips.

(5) Usurping the authentication comparison process

It must be assumed that a clone manufacturer will have access to both the camera and the print roll designs. It
15 should not be possible for the clone manufacturer to design a chip which, instead of generating an authentication code, tricks the camera into using the code generated by the duplicate authentication chip in the camera. This can be achieved by providing separate serial data Tx and Rx lines
20 for the camera and print roll authentication chips.

(6) Differential Cryptanalysis

It is important that the system adopted is secure against differential cryptanalysis. Differential
cryptanalysis is a well known technique where pairs of input
25 streams are generated with known differences, and the differences in the encoded streams are analyzed. A small amount (10^6 or so) of weakening could be accepted.

(7) Listening to the data flow between the camera and the print roll

30 Again a logic analyzer can be connected to the data stream between the camera and the print roll. In this way, the codes sent to the print roll, and the authentication reply, can be monitored. However, these codes are 128 bit pseudo-random numbers, which are only related by the
35 encoding algorithm in the authentication chips. This is essentially a known plaintext attack, which is less powerful

than a chosen plaintext attack.

(8) Direct viewing of chip operation

If the chip operation could be directly viewed using an STM or an electron beam, the authentication codes could be recorded as they are read from the internal non-volatile memory and loaded into internal registers on the chip.

(9) Direct viewing of the non-volatile memory

If the chip were sliced to that the floating gates of the Flash memory were exposed, without discharging them, then the authentication key could probably be viewed directly using an STM. However, slicing the chip to this level without discharging the gates is difficult. Using wet etching, plasma etching, ion milling, or chemical mechanical polishing will almost certainly discharge the small charges present on the floating of the chip gates.

(10) Viewing I_{dd} fluctuations

Whenever the Authentication key is being read from memory, the Message Authentication Code (MAC) circuitry is also operating, obscuring the I_{dd} signal.

Also, after the code words have been programmed, a lock bit is programmed which prevents subsequent programming of the code words. This prevents detection of the code words by monitoring the difference in I_{dd} that may occur when programming over either a high or a low bit.

(11) Bribery and other industrial espionage

It is not necessary for any human to know, or to be able to find out, what the authentication numbers are. Therefore, the numbers are safe from bribery or other corruption.

There need only be one or a few machines which programs the print roll chips, and these machines could be kept secure, preventing their theft. Authentication chips may be stolen en-route to print roll factories, but this would only enable the manufacture of as many clone print rolls as there were chips stolen, which would probably not exceed a few million in any one shipment. It would not be viable for a

print roll illegal clone manufacturer to continually steal chips.

(12) Reverse engineering the authentication key generator

5 If the clone manufacture can obtain the code for the authentication key generator, then this could be reverse engineered. For maximum security, the Authentication key should be truly random. This is simply achieve by flipping a coin 128 times, and entering the key into the authentication chip programmer is a secure environment.
10 This only has to be done once.

(13) Management decision to omit authentication to save costs

Without any form of protection, illegal cloning is almost certain. However, with the patent and copyright
15 protection, the probability of illegal cloning may be reduced to say 50%.

However, this is not the only loss possible. If a clone manufacturer were to introduce clone print rolls which caused damage to the camera (eg. clogged nozzles), then the
20 loss in market acceptance, and the expense of warranty repairs, may also be significant. One form o possible secure arrangement will now be discussed. Upon insertion of a print roll, the ACP 31 interrogates a print roll chip 53 in addition to interrogating an exact replica of the chip 54
25 stored within the camera system. The print roll chip 53 is designed to be fed a print roll test code to which it applies a one way hash function to produce a resultant code that is checked by the camera processor 105 which also sends the same code to its camera authorization chip 106.

30 Turning now to Fig. 134, there is illustrated the significant steps in the authorization method of the preferred embodiment. Each Artcam is provided with a unique random identification code 710. The Artcam processor takes the identification code 710 and a current time value 711
35 from the real time clock of the Artcam processor and exclusive ORs them together 712. The result of this process

is utilised as a seed to a random number generator 714 which produces a print roll test code having 128 bits. The Artcam processor then transmits the print roll test code to the Artcam authorization chip 54 and the print roll

5 authorization chip 53 which each utilizes their internally stored key via a corresponding roll authentication unit 702 (Fig. 132) to return to the Artcam processor 31 at stage 719 the expected output values 719 for the given input value. The Artcam processor checks to values to assure they are the
10 same and accepts or rejects the print roll based on the equality of the two values.

It will be evident from the forgoing it is crucial that the key utilised by the Artcam authorization chip 54 and print roll authorization chip 53 is kept secret. As
15 previously noted, the authorization key is stored in the flash memory store 709 of the print roll authorization chip. Therefore, an attack by way of reverse engineering of the chip will lead to minimal results. One form of attack may be to monitor the chips operation utilising a
20 scanning tunneling microscope (STM) or an electron beam to monitor the authorization codes as they are read from the internal non-volatile memory and loaded into internal registers on the chip. Turning now to Fig. 135, such analysis can be circumvented by incorporated a shielding
25 metal layer 725, over the lower circuitry of the chip 687, as an extra metalization layer.

Of course, the attacker may simply chose to wet etch the metal layer 725. However, if the metal layer 725 is utilised as the ground plane for connections within the
30 chip circuitry, the metallisation layer, if removed, will result in the chip seeking to malfunction, thereby preventing reverse analysis. This means the attacker is forced to either remove the metal layer and lay new ground connections or to mask the metal layer before removal.
35 Masking of the metal layer for removal is the easiest of

these two processes but will still be very difficult. In this case, the attacker must:

- (1) reverse engineer the chip to find out where the ground connections should be;
- 5 (2) create a mask corresponding to the required ground plane pattern connection;
- (3) apply a photo resist to the chip. This will be extremely difficult as the individual chip is only approximately 400 microns square. Therefore,
- 10 standard semi-conductor processes of applying a photo resist, in particular resist spin processing, cannot be utilised;
- (4) soft bake the resist;
- (5) expose the resist. This will again be difficult for
- 15 a single chip as modern lithographic equipment is designed for a wafer;
- (6) hard bake the resist; and
- (7) etch the top metallisation layer.

The process of high temperature resist baking will

20 most likely destroy the charge patterns in the non-volatile memory which holds the authentication numbers making this process fruitless.

Further, viewing the data flow in the chip can be made even more difficult by making all the connections

25 from which is possible to view the authentication numbers in the polysilicon layer.

The authentication key should be truly random, to prevent compromise by obtaining knowledge of the process used to generate the authentication key. A simple way is

30 for a trusted human to flip a coin 128 times, while entering 0 (heads) or 1 (tails) into the keyboard in a secure environment. The authentication key need only be known by the machine which programs the authentication chips (the human coin flipper will not remember it). So

35 that this machine cannot be stolen, all authentication numbers and chips should be programmed in one place, and

shipped to different print roll and Artcam manufacturing sites. Other data specific to a Artcam or print roll can be programmed at this place of manufacture.

Of course, it is necessary to ensure that the authentication key is never lost, as this would prevent the legitimate manufacture of compatible print rolls. Further, the bit pattern preferably contains clearly copyrightable material such that the attacker, in order to replicate the operation of the authorization chip 53 must also copy the bit pattern and therefore is likely to infringe copyright laws in various jurisdictions. The bit pattern is preferably the original work of an identifiable author reduced to a tangible form. For example, it could be a particular image of bits, otherwise it could be a corresponding ASCII equivalent of prose. Further, it should represent the application of knowledge, judgment, skill and labor by the author. It should not be an integral part of the chip but merely stored in the chips memory. Of course, preferably, the copyright ownership of the bit pattern resides with the print roll manufacturer. As an example, the bit pattern could be an ASCII representation of a short poem. Hence, the allocation of 512 bits should be sufficient. Although the bit pattern could be stored as ROM on the chips , as these chips already have flash memory, the smallest chip size may be achieved by implementing the bit pattern in the flash memory.

Turning now to Fig. 136, there is illustrated the storage table 730 of the Artcam authorization chip. The table includes manufacturing code, batch number and serial number and date which have an identical format to that previously described. The table 730 also includes information 731 on the print engine within the Artcam device. The information stored can include a print engine type, the DPI resolution of the printer and a printer

count of the number of prints produced by the printer device.

Further, an authentication test key 710 is provided which can randomly vary from chip to chip and is utilised
5 as the Artcam random identification code in the previously described algorithm. The 128 bit print roll authentication key 713 is also provided and is equivalent to the key stored within the print rolls. Next, the 512 bit pattern is stored followed by a 120 bit spare area
10 suitable for Artcam use.

As noted previously, the Artcam preferably includes a liquid crystal display 15 which indicates the number of prints left on the print roll stored within the Artcam. Further, the Artcam also includes a three state switch 17
15 which allows a user to switch between three standard formats C H and P (classic, HDTV and panoramic). Upon switching between the three states, the liquid crystal display 15 is updated to reflect the number of images left on the print roll if the particular format selected is
20 used.

In order to correctly operate the liquid crystal display, the Artcam processor, upon the insertion of a print roll and the passing of the authentication test reads the from the flash memory store of the print roll
25 chip 53 and determines the amount of paper left. Next, the value of the output format selection switch 17 is determined by the Artcam processor. Dividing the print length by the corresponding length of the selected output format the Artcam processor determines the number of
30 possible prints and updates the liquid crystal display 15 with the number of prints left. Upon a user changing the output format selection switch 17 the Artcam processor 31 re-calculates the number of output pictures in accordance with that format and again updates the LCD display 15.

35 The storage of process information in the printer roll table 705 (Fig. 130) also allows the Artcam device to

take advantage of changes in process and print characteristics of the print roll.

5 In particular, the pulse characteristics applied to each nozzle within the print head can be altered to take into account of changes in the process characteristics. Turning now to Fig. 137, the Artcam Processor can be adapted to run a software program stored in an ancillary memory ROM chip. The software program, a pulse profile characteriser 771 is able to read a number of variables from the printer roll. These variables include the remaining roll media on printer roll 772, the printer media type 773, the ink color viscosity 774, the ink color drop volume 775 and the ink color 776. Each of these variables are read by the pulse profile characteriser and a corresponding, most suitable pulse profile is determined in accordance with prior trial and experiment. The parameters alters the printer pulse received by each printer nozzle so as to improve the stability of ink output.

20 It will be evident that the authorization chip includes significant advances in that important and valuable information is stored on the printer chip with the print roll. This information can include process characteristics of the print roll in question in addition to information on the type of print roll and the amount of paper left in the print roll. Additionally, the print roll interface chip can provide valuable authentication information and can be constructed in a tamper proof manner. Further, a tamper resistant method of utilising the chip has been provided. The utilization of the print roll chip also allows a convenient and effective user interface to be provided for an immediate output form of Artcam device able to output multiple photographic formats whilst simultaneously able to provide an indicator of the number of photographs left in the printing device.

Print Head Unit

Turning now to Fig. 138 , there is illustrated an exploded perspective view, partly in section, of the print head unit 615 of Fig. 127.

5 The print head unit 615 is based around the print-head 44 which ejects ink drops on demand on to print media 611 so as to form an image. The print media 611 is pinched between two set of rollers comprising a first set 618, 616 and second set 617, 619.

10 The print-head 44 operates under the control of power, ground and signal lines 810 which provides power and control for the print-head 44 and are bonded by means of Tape Automated Bonding (TAB) to the surface of the print-head 44.

15 Importantly, the print-head 44 which can be constructed from a silicon wafer device suitably separated, relies upon a series of anisotropic etches 812 through the wafer having near vertical side walls. The through wafer etches 812 allow for the direct supply of ink to the print-head surface from the back of the wafer for subsequent ejection.

20 The ink is supplied to the back of the inkjet print-head 44 by means of ink-head supply unit 814. The inkjet print-head 44 has three separate rows along its surface for the supply of separate colors of ink. The ink-head supply unit 814 also includes a lid 815 for the sealing of ink channels.

25 In Figs. 139 - 142, there is illustrated various perspective views of the ink-head supply unit 814. Each of Figs. 139 - 142 illustrate only a portion of the ink head supply unit which can be constructed of indefinite length, the portions shown so as to provide exemplary details. In
30 Fig. 139 there is illustrated a bottom perspective view, Fig. 140 illustrates a top perspective view, Fig. 141 illustrates a close up bottom perspective view, partly in section, Fig. 142 illustrates a top side perspective view showing details of the ink channels, and Fig. 143
35 illustrates a top side perspective view as does Fig. 144.

 There is considerable cost advantage in forming ink-

head supply unit 814 from injection molded plastic instead of, say, micromachined silicon. The manufacturing cost of a plastic ink channel will be considerably less in volume and manufacturing is substantially easier. The design

5 illustrated in the accompanying Figures assumes a 1600 dpi three color monolithic print head, of a predetermined length. The provided flow rate calculations are for a 100mm photo printer.

The ink-head supply unit 814 contains all of the
10 required fine details. The lid 815 (Fig. 138) is permanently glued or ultrasonically welded to the ink-head supply unit 814 and provides a seal for the ink channels.

Turning to Fig 141, the cyan, magenta and yellow ink flows in through ink inlets 820-822, the magenta ink flows
15 through the throughholes 824,825 and along the magenta main channels 826,827 (Fig. 140). The cyan ink flows along cyan main channel 830 and the yellow ink flows along the yellow main channel 831. As best seen from Fig. 141, the cyan ink in the cyan main channels then flows into a cyan sub-channel
20 833. The yellow subchannel 834 similarly receiving yellow ink from the yellow main channel 831.

As best seen in Fig. 142, the magenta ink also flows from magenta main channels 826,827 through magenta
throughholes 836, 837. Returning again to Fig. 141, the
25 magenta ink flows out of the throughholes 836, 837. The magenta ink flows along first magenta subchannel e.g. 838 and then along second magenta subchannel e.g. 839 before flowing into a magenta trough 840. The magenta ink then flows through magenta vias e.g. 842 which are aligned with
30 corresponding inkjet head throughholes (e.g. 812 of Fig. 131) wherein they subsequently supply ink to inkjet nozzles for printing out.

Similarly, the cyan ink within the cyan subchannel 833 flows into a cyan pit area 849 which supplies ink two cyan
35 vias 843, 844. Similarly, the yellow subchannel 834 supplies yellow pit area 46 which in turn supplies yellow

vias 847, 848.

As seen in Fig. 142, the print-head is designed to be received within print-head slot 850 with the various vias e.g. 851 aligned with corresponding through holes eg. 851 in the print-head wafer.

Returning to Fig. 138, care must be taken to provide adequate ink flow to the entire print-head chip 44, while satisfying the constraints of an injection moulding process. The size of the ink through wafer holes 812 at the back of the print head chip is approximately $100\mu\text{m} \times 50\mu\text{m}$, and the spacing between through holes carrying different colors of ink is approximately $170\mu\text{m}$. While features of this size can readily be molded in plastic (compact discs have micron sized features), ideally the wall height must not exceed a few times the wall thickness so as to maintain adequate stiffness. The preferred embodiment overcomes these problems by using hierarchy of progressively smaller ink channels.

In Fig. 143, there is illustrated a small portion of the surface of the print-head 44. The surface is divided into 3 series of nozzles comprising the cyan series 871, the magenta series 872 and the yellow series 873. Each series of nozzles is further divided into two rows eg. 875, 876 with the print-head 44 having a series of bond pads 878 for bonding of power and control signals.

The print head is preferably constructed in accordance with a large number of different forms of ink jet invented for uses including Artcam devices. A full list of the different invented ink jet types is as set out in the associated Australian Provisional Patent Applications as set out appendix A attached hereto, the applications being filed on July 15, 1997 by the present applicant. In particular, the present embodiment assumes the ink jet as set out in associated Australian Provisional Patent Application number PO8064 entitled "Image Creation Method and Apparatus (IJ30)" has been utilised.

The print-head nozzles include the ink supply channels 880, equivalent to anisotropic etch hole 812 of Fig. 138. The ink flows from the back of the wafer through supply channel 881 and in turn through the filter grill 882 to ink nozzle chambers eg. 883. The operation of the nozzle chamber 883 and print-head 44 (Fig. 1) is, as mentioned previously, described in the abovementioned patent specification.

Ink Channel Fluid Flow Analysis

Turning now to an analysis of the ink flow, the main ink channels 826, 827, 830, 831 (Fig.139, Fig. 140) are around 1mm x 1mm, and supply all of the nozzles of one color. The sub-channels 833, 834, 838, 839 (Fig. 141) are around 200 μ m x 100 μ m and supply about 25 inkjet nozzles each. The print head through holes 843, 844, 847, 848 and wafer through holes eg. 881 (Fig. 143) are 100 μ m x 50 μ m and, supply 3 nozzles at each side of the print head through holes. Each nozzle filter 882 has 8 slits, each with an area of 20 μ m x 2 μ m and supplies a single nozzle.

An analysis has been conducted of the pressure requirements of an ink jet printer constructed as described. The analysis is for a 1,600 dpi three color process print head for photograph printing. The print width was 100 mm which gives 6,250 nozzles for each color, giving a total of 18,750 nozzles.

The maximum ink flow rate required in various channels for full black printing is important. It determines the pressure drop along the ink channels, and therefore whether the print head will stay filled by the surface tension forces alone, or, if not, the ink pressure that is required to keep the print head full.

To calculate the pressure drop, a drop volume of 2.5 pl for 1,600 dpi operation was utilized. While the nozzles may be capable of operating at a higher rate, the chosen drop repetition rate is 5 kHz which is suitable to print a 150 mm

long photograph in an little under 2 seconds. Thus, the print head, in the extreme case, has a 18,750 nozzles, all printing a maximum of 5,000 drops per second. This ink flow is distributed over the hierarchy of ink channels. Each ink channel effectively supplies a fixed number of nozzles when all nozzles are printing.

The pressure drop Δp was calculated according to the Darcy-Weisbach formula:

$$\Delta p = \frac{\rho U^2 f L}{2D}$$

Where ρ is the density of the ink, U is the average flow velocity, L is the length, D is the hydraulic diameter, and f is a dimensionless friction factor calculated as follows:

$$f = \frac{k}{Re}$$

Where Re is the Reynolds number and k is a dimensionless friction coefficient dependent upon the cross section of the channel calculated as follows:

$$Re = \frac{UD}{\nu}$$

Where ν is the kinematic viscosity of the ink.

For a rectangular cross section, k can be approximated by:

$$k = \frac{64}{\frac{2}{3} + \frac{11b}{24a} \left(2 - \frac{b}{a} \right) + \frac{11b}{24a}}$$

Where a is the longest side of the rectangular cross section, and b is the shortest side. The hydraulic diameter D for a rectangular cross section is given by:

$$D = \frac{2ab}{a + b}$$

Ink is drawn off the main ink channels at 250 points along the length of the channels. The ink velocity falls linearly from the start of the channel to zero at the end of the channel, so the average flow velocity U is half of the maximum flow velocity. Therefore, the pressure drop along

the main ink channels is half of that calculated using the maximum flow velocity

Utilizing these formulas, the pressure drops can be calculated in accordance with the following tables:

5 Table of Ink Channel Dimensions and Pressure Drops

	Number of Items	Length	Width	Depth	Nozzles supplied	Max.ink flow at 5KHz (U)	Pressure drop Δp
Central Moulding	1	106mm	6.4mm	1.4mm	18,750	0,23ml/sec	NA
Cyan main channel (830)	1	100mm	1mm	1mm	6,250	0.16 μ l/ μ s	111 Pa
Magenta main channel (826)	2	100mm	700 μ m	700 μ m	3,125	0.16 μ l/ μ s	231 Pa
Yellow main channel (831)	1	100mm	1mm	1mm	6,250	0.16 μ l/ μ s	111 Pa
Cyan sub-channel (833)	250	1.5mm	200 μ m	100 μ m	25	0.16 μ l/ μ s	41.7 Pa
Magenta sub- channel (834) (a)	500	200 μ m	50 μ m	100 μ m	12.5	0,031 μ l/ μ s	44.5 Pa
Magenta sub- channel (838) (b)	500	400 μ m	100 μ m	200 μ m	12.5	0.031 μ l/ μ s	5.6 Pa
Yellow sub- channel (834)	250	1.5mm	200 μ m	100 μ m	25	0.016 μ l/ μ s	41.7 Pa
Cyan pit (842)	250	200 μ m	100 μ m	300 μ m	25	0.010 μ l/ μ s	3.2 Pa
Magenta through (840)	500	200 μ m	50 μ m	200 μ m	12.5	0.016 μ l/ μ s	18.0 Pa
Yellow pit (846)	250	200 μ m	100 μ m	300 μ m	25	0.010 μ l/ μ s	3.2 Pa
Cyan via (843)	500	100 μ m	50 μ m	100 μ m	12.5	0.031 μ l/ μ s	22.3 Pa
Magenta via (842)	500	100 μ m	50 μ m	100 μ m	12.5	0.031 μ l/ μ s	22.3 Pa
Yellow via	500	100 μ m	50 μ m	100 μ m	12.5	0.031 μ l/ μ s	22.3 Pa
Magenta through hole (837)	500	200 μ m	500 μ m	100 μ m	12.5	0.0031 μ l/ μ s	0.87 Pa
Chip slot	1	100mm	730 μ m	625	18,750	NA	NA
Print head through holes (881) (in the chip substrate)	1500	600 μ	100 μ m	50 μ m	12.5	0.052 μ l/ μ s	133 Pa
Print head channel segments (on chip front)	1,000/ color	50 μ m	60 μ m	20 μ m	3.125	0.049 μ l/ μ s	62.8 Pa

Filter Slits (on entrance to nozzle chamber) (882)	8 per nozzle	2 μ m	2 μ m	20 μ m	0.125	0.039 μ l/ μ s	251 Pa
Nozzle chamber (on chip front) (883)	1 per nozzle	70 μ m	30 μ m	20 μ m	1	0.021 μ l/ μ s	75.4 Pa

The total pressure drop from the ink inlet to the nozzle is therefore approximately 701Pa for cyan and yellow, and 845 Pa for magenta. This is less than 1% of atmospheric pressure. Of course, when the image printed is less than full black, the ink flow (and therefore the pressure drop) is reduced from these values.

Making the Mould for the Ink-head Supply Unit

The ink head supply unit 14 (Fig. 1) has features as small as 50 μ and a length of 106mm. It is impractical to machine the injection moulding tools in the conventional manner. However, even though the overall shape may be complex, there are no complex curves required. The injection moulding tools can be made using conventional milling for the main ink channels and other millimeter scale features, with a lithographically fabricated inset for the fine features. A LIGA process can be used for the inset.

A single injection moulding tool could readily have 50 or more cavities. Most of the tool complexity is in the inset.

Turning to Fig. 138, the printing system is constructed via moulding ink supply unit 814 and lid 815 together and sealing them together as previously described. Subsequently print-head 44 is placed in its corresponding slot 850. Adhesive sealing strips 852, 853 are placed over the magenta main channels so to ensure they are properly sealed. The Tape Automated Bonding (TAB) strip 810 is then connected to the inkjet print-head 44 with the tab bonding wires running in the cavity 855. As can best be seen from Fig 138, 139 and 144, aperture slots are 855 - 862 are

provided for the snap in insertion of rollers. The slots provided for the "clipping in" of the rollers with a small degree of play subsequently being provided for simple rotation of the rollers.

5 In Figs. 145 - 149, there are illustrated various perspective views of the internal portions of a finally assembled Artcam device with devices appropriately numbered.

- 10 ◦ Fig. 145 illustrates a top side perspective view of the internal portions of an Artcam camera, showing the parts flattened out;
- Fig. 146 illustrates a bottom side perspective view of the internal portions of an Artcam camera, showing the parts flattened out;
- 15 ◦ Fig. 147 illustrates a first top side perspective view of the internal portions of an Artcam camera, showing the parts as encased in an Artcam;
- Fig. 148 illustrates a second top side perspective view of the internal portions of an Artcam camera, showing the parts as encased in an Artcam;
- 20 ◦ Fig. 149 illustrates a second top side perspective view of the internal portions of an Artcam camera, showing the parts as encased in an Artcam;

Postcard Print Rolls

25 Turning now to Fig. 150, in one form of the preferred embodiment, the output printer paper 11 can, on the side that is not to receive the printed image, contain a number of pre-printed "postcard" formatted backing portions 885. The postcard formatted sections 885 can include prepaid postage "stamps" 886 which can comprise a printed
30 authorization from the relevant postage authority within whose jurisdiction the print roll is to be sold or utilised. By agreement with the relevant jurisdictional postal authority, the print rolls can be made available having different postages. This is especially convenient where
35 overseas travelers are in a local jurisdiction and wishing

to send a number of postcards to their home country.
Further, an address format portion 887 is provided for the
writing of address dispatch details in the usual form of a
postcard. Finally, a message area 887 is provided for the
5 writing of a personalized information.

Turning now to Fig. 150 and Fig. 151, the operation of
the camera device is such that when a series of images 890-
892 is printed on a first surface of the print roll, the
corresponding backing surface is that illustrated in Fig.
10 150. Hence, as each image eg. 891 is printed by the camera,
the back of the image has a ready made postcard 885 which
can be immediately dispatched at the nearest post office box
within the jurisdiction. In this way, personalized
postcards can be created.

15 It would be evident that when utilising the postcard
system as illustrated in Fig. 151 and Fig. 152 only
predetermined image sizes are possible as the
synchronization between the backing postcard portion 885 and
the front image 891 must be maintained. This can be
20 achieved by utilising the memory portions of the
authentication chip stored within the print roll to store
details of the length of each postcard backing format sheet
885. This can be achieved by either having each postcard
the same size or by storing each size within the print rolls
25 on-board print chip memory.

The Artcam camera control system can ensure that, when
utilising a print roll having pre-formatted postcards, that
the printer roll is utilised only to print images such that
each image will be on a postcard boundary. Of course, a
30 degree of "play" can be provided by providing boarder
regions at the edges of each photograph which can account
for slight misalignment.

Turning now to Fig. 152, it will be evident that
postcard rolls can be pre-purchased by a camera user when
35 traveling within a particular jurisdiction where they are
available. The postcard roll can, on its external surface,

have printed information including country of purchase, the amount of postage on each postcard, the format of each postcard (for example being C,H or P or a combination of these image modes), the countries that it is suitable for
5 use with and the postage expiry date after which the postage is no longer guaranteed to be sufficient can also be provided.

Hence, a user of the camera device can produce a postcard for dispatch in the mail by utilising their hand
10 held camera to point at a relevant scene and taking a picture having the image on one surface and the pre-paid postcard details on the other. Subsequently, the postcard can be addressed and a short message written on the postcard before its immediate dispatch in the mail.

15 It would be appreciated by a person skilled in the art that numerous variations and/or modifications may be made to the present invention as shown in the specific embodiment without departing from the spirit or scope of the invention as broadly described. The present embodiment is, therefore,
20 to be considered in all respects to be illustrative and not restrictive.

The present provisional is one of a series of Australian Provisional Patent Applications which relate to a new form of technology for the production of images. These
25 Australian Provisional Patent Applications encompass a broad range of fields and as such, the present provisional is best viewed in the overall context of the development of this new form of technology. Appendix A attached hereto sets out the details of each of the series of Australian Provisional
30 Patent Applications filed on 15 July, 1997 by the present applicant, and, to the extent necessary, the associated Australian Provisional Patent Applications are hereby incorporated by cross-reference.

The present provisional further relates to research
35 conducted by the inventor in the field of ink jet technology. This research has resulted in the creation of a

large number of different classes or types of new ink jets suitable for use with the present embodiment. It would be evident to those skilled in the art, after examination of the present specification and the materials as set out in

5 the associated appendices to introduce a large number of modifications to the present embodiment in accordance with pressing economic requirements. A listing of the different variables is provided in the attached appendix B to the present specification.

10 The present provisional further relates to research conducted by the inventor in the field of ink jet technology. A number of other new ink jet designs have been set out specifically in brief form in the attached appendix C. It would be evident to those skilled in the art, after

15 examination of the present specification and the materials as set out in the associated appendix C to introduce a large number of modifications to the present embodiment in accordance with pressing economic requirements.

We Claim:

1. A digital camera system comprising:
a sensing means for sensing an image;
modification means for modifying said
5 sensed image in accordance with modification instructions
input into said camera; and
an output means for outputting said modified
image;
wherein said modification means includes a
10 series of processing elements arranged around a central
crossbar switch.
2. A digital camera as claimed in claim 1 wherein
said processing elements include an Arithmetic Logic Unit
(ALU) acting under the control of a microcode store
15 wherein said microcode store comprises a writeable control
store.
3. digital camera as claimed in any previous claim
wherein said processing elements include an internal input
and output FIFO for storing pixel data utilized by said
20 processing elements.
4. A digital camera system as claimed in any
previous claim wherein said modification means is
interconnected to a read and write FIFO for reading and
writing pixel data of images to said modification means.
- 25 5. A digital camera as claimed in any previous
claim wherein said processing elements are arranged in a
ring and each element is also separately connected to its
nearest neighbours.
6. A digital camera as claimed in any of claims 2
30 to 5 wherein said ALU accepts a series of inputs
interconnected via an internal crossbar switch to a series
of core processing units within said ALU.
7. A digital camera as claimed in claim 6 wherein
said core processing units include at least one one of a
35 multiplier, an adder and a barrel shifter.

8. A digital camera as claimed in claim 6 or 7 wherein said ALU includes a number of internal registers for the storage of temporary data.

5 9. A digital camera as claimed in any previous claim wherein said processing elements are further connected to a common data bus for the transfer of pixel data to said processing elements.

10 10. A digital camera as claimed in claim 9 where said data bus is interconnected to a data cache which acts as an intermediate cache between said processing elements and a memory store for storing said images.

15

Dated this 11th day of August 1997

Silverbrook Research Pty Ltd

By their Patent Attorneys

GRIFFITH HACK

Abstract

A digital camera system comprising a sensing means for sensing an image; modification means for modifying the sensed image in accordance with modification instructions
5 input into the camera; and an output means for outputting the modified image; wherein the modification means includes a series of processing elements arranged around a central crossbar switch. The processing elements include an Arithmetic Logic Unit (ALU) acting under the control of
10 a microcode store wherein the microcode store comprises a writeable control store. The processing elements can include an internal input and output FIFO for storing pixel data utilized by the processing elements and the modification means is interconnected to a read and write
15 FIFO for reading and writing pixel data of images to the modification means. Each of the processing elements can be arranged in a ring and each element is also separately connected to its nearest neighbours. The ALU accepts a series of inputs interconnected via an internal crossbar
20 switch to a series of core processing units within the ALU and includes a number of internal registers for the storage of temporary data. The core processing units can include at least one one of a multiplier, an adder and a barrel shifter. The processing elements are further
25 connected to a common data bus for the transfer of pixel data to the processing elements and the data bus is interconnected to a data cache which acts as an intermediate cache between the processing elements and a memory store for storing the images.

Appendix A – Related Australian Provisional Patent Applications

The present provisional is integrally related to a series of Australian Provisional Patent Applications filed by the present Applicant on 15 July, 1997 and which together relate to a new image processing system which presents a large number of significant advances in a number of technological fields. These fields include, but are not limited to those set out in the following table:

- Camera technologies
- Display technologies
- Image processing
- Ink Jet printing technology
- Semiconductor fabrication technology
- Micro Electro Mechanical Systems (MEMS)
- VLSI and ULSI fabrication including Thin Film Technology
- Magnetics
- Fluid dynamics
- Precision engineering
- Plastics molding
- Materials science
- Digital systems architecture
- Fluid Dynamics
- Precision Engineering
- Non-impact printing technologies
- Mechanical and stress analysis
- Ink Chemistry
- Electronics
- Electrostatics

Naturally with such a large number of significant advances, it is necessary to read this Application with its associated Australian Provisional Patent Applications to gain a thorough understanding of the operation of these technologies. The following tables set out a full list of the associated Australian Provisional Patent Applications filed on 15 July, 1997 by the present applicant which should be referred to in obtaining a full understanding of the operation of the present invention:

Ink Jet Printing

A large number of new forms of ink jet printers have been developed to facilitate alternative ink jet technologies for the image processing system. Australian Provisional Patent Applications relating to these ink jets include:

- Image Creation Method and Apparatus (IJ01)
- Image Creation Method and Apparatus (IJ02)
- Image Creation Method and Apparatus (IJ03)
- Image Creation Method and Apparatus (IJ04)
- Image Creation Method and Apparatus (IJ05)
- Image Creation Method and Apparatus (IJ06)
- Image Creation Method and Apparatus (IJ07)
- Image Creation Method and Apparatus (IJ08)
- Image Creation Method and Apparatus (IJ09)
- Image Creation Method and Apparatus (IJ10)
- Image Creation Method and Apparatus (IJ11)
- Image Creation Method and Apparatus (IJ12)
- Image Creation Method and Apparatus (IJ13)
- Image Creation Method and Apparatus (IJ14)

Image Creation Method and Apparatus (IJ15)
 Image Creation Method and Apparatus (IJ16)
 Image Creation Method and Apparatus (IJ17)
 Image Creation Method and Apparatus (IJ18)
 Image Creation Method and Apparatus (IJ19)
 Image Creation Method and Apparatus (IJ20)
 Image Creation Method and Apparatus (IJ21)
 Image Creation Method and Apparatus (IJ22)
 Image Creation Method and Apparatus (IJ23)
 Image Creation Method and Apparatus (IJ24)
 Image Creation Method and Apparatus (IJ25)
 Image Creation Method and Apparatus (IJ26)
 Image Creation Method and Apparatus (IJ27)
 Image Creation Method and Apparatus (IJ28)
 Image Creation Method and Apparatus (IJ29)
 Image Creation Method and Apparatus (IJ30)
 Supply Method and Apparatus (F1)
 Supply Method and Apparatus (F2)

Ink Jet Manufacturing

Significant developments have occurred in the field of ink jet print head construction. These advances are included in the following Australian Provisional Patent Applications.

A Method of Manufacture of an Image Creation Apparatus (IJM01)
 A Method of Manufacture of an Image Creation Apparatus (IJM02)
 A Method of Manufacture of an Image Creation Apparatus (IJM03)
 A Method of Manufacture of an Image Creation Apparatus (IJM04)
 A Method of Manufacture of an Image Creation Apparatus (IJM05)
 A Method of Manufacture of an Image Creation Apparatus (IJM06)
 A Method of Manufacture of an Image Creation Apparatus (IJM07)
 A Method of Manufacture of an Image Creation Apparatus (IJM08)
 A Method of Manufacture of an Image Creation Apparatus (IJM09)
 A Method of Manufacture of an Image Creation Apparatus (IJM10)
 A Method of Manufacture of an Image Creation Apparatus (IJM11)
 A Method of Manufacture of an Image Creation Apparatus (IJM12)
 A Method of Manufacture of an Image Creation Apparatus (IJM13)
 A Method of Manufacture of an Image Creation Apparatus (IJM14)
 A Method of Manufacture of an Image Creation Apparatus (IJM15)
 A Method of Manufacture of an Image Creation Apparatus (IJM16)
 A Method of Manufacture of an Image Creation Apparatus (IJM17)
 A Method of Manufacture of an Image Creation Apparatus (IJM18)
 A Method of Manufacture of an Image Creation Apparatus (IJM19)
 A Method of Manufacture of an Image Creation Apparatus (IJM20)
 A Method of Manufacture of an Image Creation Apparatus (IJM21)
 A Method of Manufacture of an Image Creation Apparatus (IJM22)
 A Method of Manufacture of an Image Creation Apparatus (IJM23)
 A Method of Manufacture of an Image Creation Apparatus (IJM24)
 A Method of Manufacture of an Image Creation Apparatus (IJM25)
 A Method of Manufacture of an Image Creation Apparatus (IJM26)
 A Method of Manufacture of an Image Creation Apparatus (IJM27)
 A Method of Manufacture of an Image Creation Apparatus (IJM28)
 A Method of Manufacture of an Image Creation Apparatus (IJM29)
 A Method of Manufacture of an Image Creation Apparatus (IJM30)

MEMS Technology

The following application relate to Micro Electro-Mechanical Systems technologies:

A device (MEMS01)
 A device (MEMS02)
 A device (MEMS03)
 A device (MEMS04)
 A device (MEMS05)
 A device (MEMS06)
 A device (MEMS07)
 A device (MEMS08)
 A device (MEMS09)
 A device (MEMS10)

Artcam Technologies

The following Australian Provisional Patent Applications relate to the a new field of image processing technology known as Artcam.

Image Processing Method and Apparatus (ART01)
 Image Processing Method and Apparatus (ART02)
 Image Processing Method and Apparatus (ART03)
 Image Processing Method and Apparatus (ART05)
 Image Processing Method and Apparatus (ART06)
 Media Device (ART07)
 Image Processing Method and Apparatus (ART08)
 Image Processing Method and Apparatus (ART09)
 Image Processing Method and Apparatus (ART10)
 Image Processing Method and Apparatus (ART11)
 Image Processing Method and Apparatus (ART12)
 Media Device (ART13)
 Image Processing Method and Apparatus (ART14)
 Media Device (ART15)
 Media Device (ART16)
 Media Device (ART17)
 Media Device (ART18)
 Data Processing Method and Apparatus (ART19)
 Data Processing Method and Apparatus (ART20)
 Media Processing Method and Apparatus (ART21)
 Image Processing Method and Apparatus (ART22)
 Image Processing Method and Apparatus (ART23)
 Image Processing Method and Apparatus (ART24)
 Image Processing Method and Apparatus (ART25)
 Image Processing Method and Apparatus (ART26)
 Image Processing Method and Apparatus (ART27)
 Data Processing Method and Apparatus (ART29)
 Data Processing Method and Apparatus (ART32)
 Image Processing Method and Apparatus (ART33)
 Sensor Creation Method and Apparatus (ART36)
 Data Processing Method and Apparatus (ART37)
 Data Processing Method and Apparatus (ART38)
 Data Processing Method and Apparatus (ART39)
 Data Processing Method and Apparatus (ART40)
 Data Processing Method and Apparatus (ART43)
 Data Processing Method and Apparatus (ART44)
 Data Processing Method and Apparatus (ART45)
 Data Processing Method and Apparatus (ART46)

Data Processing Method and Apparatus (ART50)
Data Processing Method and Apparatus (ART51)
Data Processing Method and Apparatus (ART52)
Image Processing Method and Apparatus (ART53)
Image Processing Method and Apparatus (ART54)
Image Processing Method and Apparatus (ART56)

Appendix B – Ink Jet Types

Physical Variables

The present invention utilises an ink jet printer. In the construction of ink jet printing devices many trade offs can be made and many factors will influence the final form of any particular ink jet printer device. For an example of the many different factors utilized in the construction of ink jet printers reference is made to the annual proceedings of the IS & T International Congress on Advances in Non-Impact Printing Technologies.

The present invention, in utilizing an ink jet printer, presents a number of different alternatives in construction, many of which have been independently invented by Kia Silverbrook. As a result of these inventions, it is considered that a large number of choices could be made by the person skilled in the art as a result of this and previous disclosures in the field of construction of ink jet print heads. These independent variations can be grouped in accordance with the following table:

Physical Variable	Number of Values
Actuator mechanism	18 entries
Basic operation mode	7 entries
Auxiliary mechanism	8 entries
Actuator amplification or modification method	13 entries
Actuator motion	14 entries
Nozzle refill method	4 entries
Nozzle plate construction	8 entries
Drop ejection direction	5 entries
Ink type	6 entries

Obviously, selection of suitable values from the above table will result in a plethora of different possible designs of which a large number will be commercially viable. A number of these designs have been invented by other parties and many have been indicated in the table below. In addition, Kia Silverbrook has previously invented one form of ink jet printer, hereinafter known as LIFT and disclosed in a series of applications filed as Australian Provisional Patent Applications on 12 April, 1995 including application PN2308 entitled "A Liquid Ink Fault Tolerant Ink Mechanism".

The accompanying appendix C sets out a selection of the main ink jet configurations involving novel arrangements of the above variables. These ink jet configurations have been denoted IJ01 to IJ30.

For further discussion of the operation of each of the ink jets IJ01 to IJ30, reference is made to the corresponding Australian Provisional Patent Applications filed concurrently herewith. Appendix A, attached to the present specification, contains a list of all the Australian Provisional Patent Applications filed concurrently herewith as part of this project. Each of the ink jet patents are denoted with the "IJ" number in their title. Each of the IJ01 to IJ30 examples can be made into ink jet print heads with superior characteristics to any currently available ink jet technology.

Variations in physical variables

The following tables set out a number of variations in each physical variable in the table listed above. These tables only address drop-on-demand ink jet technologies. Continuous ink jet technologies are not specifically addressed.

Actuator mechanism (applied only to selected nozzles)

Actuator Mechanism	Description	Advantages	Disadvantages	Examples
Thermal bubble	<p>An electrothermal heater heats the ink to above boiling point, transferring significant heat to the aqueous ink. A bubble nucleates and quickly forms, expelling the ink.</p> <p>The efficiency of the process is low, with typically less than 0.05% of the electrical energy being transformed into kinetic energy of the drop.</p>	<ul style="list-style-type: none"> High force generated Simple construction No moving parts Fast operation Small chip area required for actuator 	<ul style="list-style-type: none"> High power Ink carrier limited to water Low efficiency High temperatures required High mechanical stress Unusual materials required Large drive transistors Cavitation causes actuator failure Kogation reduces bubble formation Large print heads are difficult to fabricate 	<ul style="list-style-type: none"> Canon Bubblejet 1979 Endo et al GB patent 2,007,162 Xerox heater-in-pit 1990 Hawkins et al USP 4,899,181 Hewlett-Packard TIJ 1982 Vaught et al USP 4,490,728
Piezoelectric	A piezoelectric crystal such as lead lanthanum zirconate (PZT) is electrically activated, and either expands, shears, or bends to apply pressure to the ink, ejecting drops.	<ul style="list-style-type: none"> Low power consumption Many ink types can be used Fast operation High efficiency 	<ul style="list-style-type: none"> Very large area required for actuator Difficult to integrate with electronics High voltage drive transistors required Full pagewidth print heads impractical due to actuator size Requires electrical poling in high field strengths during manufacture 	<ul style="list-style-type: none"> Kyser et al USP 3,946,398 Zoltan USP 3,683,212 1973 Stemme USP 3,747,120 Epson Stylus Tektronix IJ04
Electro-strictive	An electric field is used to activate electrostriction in relaxor materials such as lead lanthanum zirconate titanate (PLZT) or lead magnesium niobate (PMN).	<ul style="list-style-type: none"> Low power consumption Many ink types can be used Low thermal expansion Electric field strength required (approx. 3.5 V/μm) can be generated without difficulty Does not require electrical poling 	<ul style="list-style-type: none"> Low maximum strain (approx. 0.01%) Large area required for actuator due to low strain Response speed is marginal ($\sim 10 \mu$s) High voltage drive transistors required Full pagewidth print heads impractical due to actuator size 	IJ04
Ferroelectric	An electric field is used to induce a phase transition between the antiferroelectric (AFE) and ferroelectric (FE) phase. Perovskite materials such as tin modified lead lanthanum zirconate titanate (PLZSnT) exhibit large strains of up to 1% associated with the AFE to FE phase transition.	<ul style="list-style-type: none"> Low power consumption Many ink types can be used Fast operation ($< 1 \mu$s) Relatively high longitudinal strain High efficiency Electric field strength of around 3 V/μm can be readily provided 	<ul style="list-style-type: none"> Difficult to integrate with electronics Unusual materials such as PLZSnT are required Actuators require a large area 	IJ04

Electrostatic plates	Conductive plates are separated by a compressible or fluid dielectric (usually air). Upon application of a voltage, the plates attract each other and displace ink, causing drop ejection. The conductive plates may be in a comb or honeycomb structure, or stacked to increase the surface area and therefore the force.	<ul style="list-style-type: none"> Low power consumption Many ink types can be used Fast operation 	<ul style="list-style-type: none"> Difficult to operate electrostatic devices in an aqueous environment The electrostatic actuator will normally need to be separated from the ink Very large area required to achieve high forces High voltage drive transistors may be required Full pagewidth print heads are not competitive due to actuator size 	<ul style="list-style-type: none"> IJ02, IJ04
Electrostatic pull on ink	A strong electric field is applied to the ink, whereupon electrostatic attraction accelerates the ink towards the print medium.	<ul style="list-style-type: none"> Low current consumption Low temperature 	<ul style="list-style-type: none"> High voltage required May be damaged by sparks due to air breakdown Required field strength increases as the drop size decreases High voltage drive transistors required Electrostatic field attracts dust 	<ul style="list-style-type: none"> 1989 Saito et al, USP 4,799,068 1989 Miura et al, USP 4,810,954
Permanent magnet electro-magnetic	An electromagnet directly attracts a permanent magnet, displacing ink and causing drop ejection. Rare earth magnets with a field strength around 1 Tesla can be used. Examples are: Samarium Cobalt (SaCo) and magnetic materials in the neodymium iron boron family (NdFeB, NdDyFeBNb, NdDyFeB, etc)	<ul style="list-style-type: none"> Low power consumption Many ink types can be used Fast operation High efficiency Easy extension from single nozzles to pagewidth print heads 	<ul style="list-style-type: none"> Complex fabrication Permanent magnetic material such as Neodymium Iron Boron (NdFeB) required High local currents required Copper metallization should be used for long electromigration lifetime and low resistivity Pigmented inks are usually infeasible Operating temperature limited to the Curie temperature (around 540 K) 	<ul style="list-style-type: none"> IJ07, IJ10
Soft magnetic core electro-magnetic	A solenoid induced a magnetic field in a soft magnetic core or yoke fabricated from a ferrous material such as nickel iron (NiFe). Typically, the soft magnetic material is in two parts, which are normally held apart by a spring. When the solenoid is actuated, the two parts attract, displacing the ink.	<ul style="list-style-type: none"> Low power consumption Many ink types can be used Fast operation High efficiency Easy extension from single nozzles to pagewidth print heads 	<ul style="list-style-type: none"> Complex fabrication Unusual materials such as nickel iron (NiFe) are required High local currents required Copper metallization should be used for long electromigration lifetime and low resistivity Pigmented inks are usually infeasible 	<ul style="list-style-type: none"> IJ01, IJ05, IJ08, IJ10 IJ12, IJ14, IJ15, IJ17

Magnetic Lorenz force	<p>The Lorenz force acting on a current carrying wire in a magnetic field is utilized.</p> <p>This allows the magnetic field to be supplied externally to the print head, for example with rare earth permanent magnets.</p> <p>Only the current carrying wire need be fabricated on the print-head, simplifying materials requirements.</p>	<ul style="list-style-type: none"> ◊ Low power consumption ◊ Many ink types can be used ◊ Fast operation ◊ High efficiency ◊ Easy extension from single nozzles to pagewidth print heads 	<ul style="list-style-type: none"> ◊ Force acts as a twisting motion ◊ Typically, only a quarter of the solenoid length provides force in a useful direction ◊ High local currents required ◊ Copper metallization should be used for long electromigration lifetime and low resistivity ◊ Pigmented inks are usually infeasible 	<ul style="list-style-type: none"> ◊ IJ06, IJ11, IJ13, IJ16
Magneto-striction	<p>The actuator uses the giant magnetostrictive effect of materials such as Terfenol-D (an alloy of terbium, dysprosium and iron developed at the Naval Ordnance Laboratory, hence Ter-Fe-NOL). For best efficiency, the actuator should be pre-stressed to approx. 8 MPa.</p>	<ul style="list-style-type: none"> ◊ Many ink types can be used ◊ Fast operation ◊ Easy extension from single nozzles to pagewidth print heads ◊ High force is available 	<ul style="list-style-type: none"> ◊ Force acts as a twisting motion ◊ Unusual materials such as Terfenol-D are required ◊ High local currents required ◊ Copper metallization should be used for long electromigration lifetime and low resistivity ◊ Pre-stressing may be required 	<ul style="list-style-type: none"> ◊ IJ25
Surface tension reduction	<p>Ink under positive pressure is held in a nozzle by surface tension. The surface tension of the ink is reduced below the bubble threshold, causing the ink to egress from the nozzle.</p>	<ul style="list-style-type: none"> ◊ Low power consumption ◊ Simple construction ◊ No unusual materials required in fabrication ◊ High efficiency ◊ Easy extension from single nozzles to pagewidth print heads 	<ul style="list-style-type: none"> ◊ Requires supplementary force to effect drop separation ◊ Requires special ink surfactants ◊ Speed may be limited by surfactant properties 	<ul style="list-style-type: none"> ◊ LIFT
Viscosity reduction	<p>The ink viscosity is locally reduced to select which drops are to be ejected. A viscosity reduction can be achieved electrothermally with most inks, but special inks can be engineered for a 100:1 viscosity reduction.</p>	<ul style="list-style-type: none"> ◊ Simple construction ◊ No unusual materials required in fabrication ◊ Easy extension from single nozzles to pagewidth print heads 	<ul style="list-style-type: none"> ◊ Requires supplementary force to effect drop separation ◊ Requires special ink viscosity properties ◊ High speed is difficult to achieve ◊ Requires oscillating ink pressure ◊ A high temperature difference (typically 80 degrees) is required 	<ul style="list-style-type: none"> ◊ LIFT
Acoustic	<p>An acoustic wave is generated and focussed upon the drop ejection region.</p>	<ul style="list-style-type: none"> ◊ Can operate without a nozzle plate 	<ul style="list-style-type: none"> ◊ Complex drive circuitry ◊ Complex fabrication ◊ Low efficiency ◊ Poor control of drop position ◊ Poor control of drop volume 	<ul style="list-style-type: none"> ◊ 1993 Hadimioglu et al, EUP 550,192 ◊ 1993 Elrod et al, EUP 572,220

Thermoelastic bend actuator	An actuator which relies upon differential thermal expansion upon Joule heating is used.	<ul style="list-style-type: none"> ♦ Low power consumption ♦ Many ink types can be used ♦ Simple planar fabrication ♦ Small chip area required for each actuator ♦ Fast operation ♦ High efficiency ♦ CMOS compatible voltages and currents ♦ Standard MEMS processes can be used ♦ Easy extension from single nozzles to pagewidth print heads 	<ul style="list-style-type: none"> ♦ Difficult to achieve a large force, large travel, and small size simultaneously ♦ Efficient aqueous operation requires a thermal insulator on the hot arm ♦ Corrosion prevention can be difficult ♦ Pigmented inks may be infeasible, as pigment particles may jam the bend actuator 	<ul style="list-style-type: none"> ♦ IJ03, IJ09, IJ17, IJ18 ♦ IJ19, IJ20, IJ21, IJ22 ♦ IJ23, IJ24, IJ27, IJ28 ♦ IJ29, IJ30
High CTE thermoelastic actuator	<p>A material with a very high coefficient of thermal expansion (CTE) such as polytetrafluoroethylene (PTFE) is used. As high CTE materials are usually non conductive, a heater fabricated from a conductive material is incorporated. A 50 μm long PTFE bend actuator with polysilicon heater and 15 mW power input can provide 180 μN force and 10 μm deflection. Actuator motions include:</p> <ol style="list-style-type: none"> 1) Bend 2) Push 3) Buckle 4) Rotate 	<ul style="list-style-type: none"> ♦ High force can be generated ♦ Three methods of PTFE deposition are under development: chemical vapor deposition (CVD), spin coating, and evaporation ♦ PTFE is a candidate for low dielectric constant insulation in ULSI ♦ Very low power consumption ♦ Many ink types can be used ♦ Simple planar fabrication ♦ Small chip area required for each actuator ♦ Fast operation ♦ High efficiency ♦ CMOS compatible voltages and currents ♦ Easy extension from single nozzles to pagewidth print heads 	<ul style="list-style-type: none"> ♦ Requires special material (e.g. PTFE) ♦ Requires a PTFE deposition process, which is not yet standard in ULSI fabs ♦ PTFE deposition cannot be followed with high temperature (above 350 °C) processing ♦ Pigmented inks may be infeasible, as pigment particles may jam the bend actuator 	<ul style="list-style-type: none"> ♦ IJ09, IJ17, IJ18, IJ20 ♦ IJ21, IJ22, IJ23, IJ24 ♦ IJ27, IJ28, IJ29, IJ30

Conductive polymer thermoelastic actuator	<p>A polymer with a high coefficient of thermal expansion (such as PTFE) is doped with conducting substances to increase its conductivity to about 3 orders of magnitude below that of copper. The conducting polymer expands when resistively heated.</p> <p>Examples of conducting dopants include:</p> <ol style="list-style-type: none"> 1) Carbon nanotubes 2) Metal fibers 3) Conductive polymers such as doped polythiophene 4) Carbon granules 	<ul style="list-style-type: none"> ♦ High force can be generated ♦ Very low power consumption ♦ Many ink types can be used ♦ Simple planar fabrication ♦ Small chip area required for each actuator ♦ Fast operation ♦ High efficiency ♦ CMOS compatible voltages and currents ♦ Easy extension from single nozzles to pagewidth print heads 	<ul style="list-style-type: none"> ♦ Requires special materials development (High CTE conductive polymer) ♦ Requires a PTFE deposition process, which is not yet standard in ULSI fabs ♦ PTFE deposition cannot be followed with high temperature (above 350 °C) processing ♦ Evaporation and CVD deposition techniques cannot be used ♦ Pigmented inks may be infeasible, as pigment particles may jam the bend actuator 	<ul style="list-style-type: none"> ♦ IJ24
Shape memory alloy	<p>A shape memory alloy such as TiNi (also known as Nitinol - Nickel Titanium alloy developed at the Naval Ordnance Laboratory) is thermally switched between its weak martensitic state and its high stiffness austenitic state. The shape of the actuator in its martensitic state is deformed relative to the austenitic shape. The shape change causes ejection of a drop.</p>	<ul style="list-style-type: none"> ♦ High force is available (stresses of hundreds of MPa) ♦ Large strain is available (more than 3%) ♦ High corrosion resistance ♦ Simple construction ♦ Easy extension from single nozzles to pagewidth print heads ♦ Low voltage operation 	<ul style="list-style-type: none"> ♦ Fatigue limits maximum number of cycles ♦ Low strain (1%) is required to extend fatigue resistance ♦ Cycle rate limited by heat removal ♦ Requires unusual materials (TiNi) ♦ The latent heat of transformation must be provided ♦ High current operation ♦ Requires pre-stressing to distort the martensitic state 	<ul style="list-style-type: none"> ♦ IJ26
Linear Magnetic Actuator	<p>Linear magnetic actuators include the Linear Induction Actuator (LIA), Linear Permanent Magnet Synchronous Actuator (LPMSA), Linear Reluctance Synchronous Actuator (LRSA), Linear Switched Reluctance Actuator (LSRA), and the Linear Stepper Actuator (LSA).</p>	<ul style="list-style-type: none"> ♦ Linear Magnetic actuators can be constructed with high thrust, long travel, and high efficiency using planar semiconductor fabrication techniques ♦ Long actuator travel is available ♦ Medium force is available ♦ Low voltage operation 	<ul style="list-style-type: none"> ♦ Requires unusual semiconductor materials such as soft magnetic (NiFe) ♦ Some varieties also require permanent magnetic materials such as Neodymium iron boron (NdFeB) ♦ Requires complex multi-phase drive circuitry ♦ High current operation 	<ul style="list-style-type: none"> ♦ IJ12

Basic operation mode

Operational mode	Description	Advantages	Disadvantages	Examples
Actuator directly pushes ink	This is the simplest mode of operation: the actuator directly supplies sufficient kinetic energy to expel the drop. The drop must have a sufficient velocity to overcome the surface tension.	<ul style="list-style-type: none"> Simple operation No external fields required Satellite drops can be avoided if drop velocity is less than 4 m/s Can be efficient, depending upon the actuator used 	<ul style="list-style-type: none"> Drop repetition rate is usually limited to less than 10 KHz. However, this is not fundamental to the method, but is related to the refill method normally used All of the drop kinetic energy must be provided by the actuator Satellite drops usually form if drop velocity is greater than 4 m/s 	<ul style="list-style-type: none"> Thermal ink jet Piezo-electric ink jet IJ01, IJ02, IJ03, IJ04 IJ05, IJ06, IJ07, IJ09 IJ11, IJ12, IJ14, IJ16 IJ20, IJ22, IJ23, IJ24 IJ25, IJ26, IJ27, IJ28 IJ29, IJ30
Proximity	The drops to be printed are selected by some manner (e.g. thermally induced surface tension reduction of pressurized ink). Selected drops are separated from the ink in the nozzle by contact with the print medium or a transfer roller.	<ul style="list-style-type: none"> Very simple print head fabrication can be used The drop selection means does not need to provide the energy required to separate the drop from the nozzle 	<ul style="list-style-type: none"> Requires close proximity between the print head and the print media or transfer roller May require two print heads printing alternate rows of the image Monolithic color print heads are difficult 	<ul style="list-style-type: none"> LIFT
Electrostatic pull on ink	The drops to be printed are selected by some manner (e.g. thermally induced surface tension reduction of pressurized ink). Selected drops are separated from the ink in the nozzle by a strong electric field.	<ul style="list-style-type: none"> Very simple print head fabrication can be used The drop selection means does not need to provide the energy required to separate the drop from the nozzle 	<ul style="list-style-type: none"> Requires very high electrostatic field Electrostatic field for small nozzle sizes is above air breakdown Electrostatic field may attract dust 	<ul style="list-style-type: none"> LIFT
Magnetic pull on ink	The drops to be printed are selected by some manner (e.g. thermally induced surface tension reduction of pressurized ink). Selected drops are separated from the ink in the nozzle by a strong magnetic field acting on the magnetic ink.	<ul style="list-style-type: none"> Very simple print head fabrication can be used The drop selection means does not need to provide the energy required to separate the drop from the nozzle 	<ul style="list-style-type: none"> Requires magnetic ink Ink colors other than black are difficult Requires very high magnetic fields 	<ul style="list-style-type: none"> LIFT
Shutter	The actuator moves a shutter to block ink flow to the nozzle. The ink pressure is pulsed at a multiple of the drop ejection frequency.	<ul style="list-style-type: none"> High speed (>50 KHz) operation can be achieved due to reduced refill time Drop timing can be very accurate The actuator energy can be very low 	<ul style="list-style-type: none"> Moving parts are required Requires ink pressure modulator Friction and wear must be considered Stiction is possible 	<ul style="list-style-type: none"> IJ13, IJ17, IJ21
Shuttered grill	The actuator moves a shutter to block ink flow through a grill to the nozzle. The shutter movement need only be equal to the width of the grill holes.	<ul style="list-style-type: none"> Actuators with small travel can be used Actuators with small force can be used High speed (>50 KHz) operation can be achieved 	<ul style="list-style-type: none"> Moving parts are required Requires ink pressure modulator Friction and wear must be considered Stiction is possible 	<ul style="list-style-type: none"> IJ08, IJ15, IJ18, IJ19

Pulsed magnetic pull on ink pusher	A pulsed magnetic field attracts an 'ink pusher' at the drop ejection frequency. An actuator controls a catch, which prevents the ink pusher from moving when a drop is not to be ejected.	<ul style="list-style-type: none">♦ Extremely low energy operation is possible♦ No heat dissipation problems	<ul style="list-style-type: none">♦ Requires an external pulsed magnetic field♦ Requires special materials for both the actuator and the ink pusher♦ Complex construction	<ul style="list-style-type: none">♦ IJ10
---	--	---	---	--

Auxiliary mechanism (applied to all nozzles)

Auxiliary Mechanism	Description	Advantages	Disadvantages	Examples
None	The actuator directly fires the ink drop, and there is no external field or other mechanism required.	<ul style="list-style-type: none"> ◊ Simplicity of construction ◊ Simplicity of operation ◊ Small physical size 	<ul style="list-style-type: none"> ◊ Drop ejection energy must be supplied by individual nozzle actuator 	<ul style="list-style-type: none"> ◊ Most ink jets, including piezoelectric and thermal bubble. ◊ IJ01, IJ02, IJ03, IJ04 ◊ IJ05, IJ07, IJ09, IJ11 ◊ IJ12, IJ14, IJ20, IJ22 ◊ IJ23, IJ24, IJ25, IJ26 ◊ IJ27, IJ28, IJ29, IJ30
Oscillating ink pressure	The ink pressure oscillates, providing much of the drop ejection energy. The actuator selects which drops are to be fired by selectively blocking or enabling nozzles. The ink pressure oscillation may be achieved by vibrating the print head, or preferably by an actuator in the ink supply.	<ul style="list-style-type: none"> ◊ Oscillating ink pressure can provide a refill pulse, allowing higher operating speed ◊ The actuators may operate with much lower energy 	<ul style="list-style-type: none"> ◊ Requires external ink pressure oscillator ◊ Ink pressure phase and amplitude must be carefully controlled 	<ul style="list-style-type: none"> ◊ Viscous LIFT ◊ IJ08, IJ13, IJ15, IJ17 ◊ IJ18, IJ19, IJ21
Media proximity	The print head is placed in close proximity to the print medium. Selected drops protrude from the print head further than unselected drops, and contact the print medium. The drop soaks into the medium fast enough to cause drop separation.	<ul style="list-style-type: none"> ◊ Low power ◊ High accuracy ◊ Simple print head construction 	<ul style="list-style-type: none"> ◊ Precision assembly required ◊ Paper fibers may cause problems ◊ Cannot print on rough substrates 	<ul style="list-style-type: none"> ◊ Proximity LIFT
Transfer roller	Drops are printed to a transfer roller instead of straight to the print medium. A transfer roller can also be used for proximity drop separation.	<ul style="list-style-type: none"> ◊ High accuracy ◊ Wide range of print substrates can be used ◊ Ink can be dried on the transfer roller 	<ul style="list-style-type: none"> ◊ Bulky ◊ Expensive ◊ Complex construction 	<ul style="list-style-type: none"> ◊ Proximity LIFT ◊ Tektronix hot melt piezo-electric ink jet ◊ Any of the IJ series
Electrostatic	An electric field is used to accelerate selected drops towards the print medium.	<ul style="list-style-type: none"> ◊ Low power ◊ Simple print head construction 	<ul style="list-style-type: none"> ◊ Field strength required for separation of small drops is near or above air breakdown 	<ul style="list-style-type: none"> ◊ LIFT
Direct magnetic field	An magnetic field is used to accelerate selected drops of magnetic ink towards the print medium.	<ul style="list-style-type: none"> ◊ Low power ◊ Simple print head construction 	<ul style="list-style-type: none"> ◊ Requires magnetic ink ◊ Requires strong magnetic field 	<ul style="list-style-type: none"> ◊ LIFT
Cross magnetic field	The print head is placed in a constant magnetic field. The Lorentz force in a current carrying wire is used to move the actuator.	<ul style="list-style-type: none"> ◊ Does not require magnetic materials to be integrated in the print head manufacturing process 	<ul style="list-style-type: none"> ◊ Requires external magnet ◊ Current densities may be high, resulting in electromigration problems 	<ul style="list-style-type: none"> ◊ IJ06, IJ16
Pulsed magnetic field	A pulsed magnetic field is used to cyclically attract a paddle which pushes on the ink. A small actuator moves a catch which selectively prevents the paddle from moving.	<ul style="list-style-type: none"> ◊ Very low power operation is possible ◊ Small print head size 	<ul style="list-style-type: none"> ◊ Complex print head construction ◊ Magnetic materials required in print head 	<ul style="list-style-type: none"> ◊ IJ10

Actuator amplification or modification method

Actuator amplification	Description	Advantages	Disadvantages	Examples
None	No actuator mechanical amplification is used. The actuator directly drives the drop ejection process.	<ul style="list-style-type: none"> Operational simplicity 	<ul style="list-style-type: none"> Many actuator mechanisms have insufficient travel, or insufficient force, to efficiently drive the drop ejection process 	<ul style="list-style-type: none"> Thermal Bubble Ink Jet IJ01, IJ02, IJ06, IJ07 IJ16, IJ25, IJ26
Differential expansion bend actuator	An actuator material expands more on one side than on the other. The expansion may be thermal, piezoelectric, magnetostrictive, or other mechanism. The bend actuator converts a high force low travel actuator mechanism to high travel, lower force mechanism.	<ul style="list-style-type: none"> Provides greater travel in a reduced print head area 	<ul style="list-style-type: none"> High stresses are involved Care must be taken that the materials do not delaminate 	<ul style="list-style-type: none"> Piezoelectric IJ03, IJ09, IJ17, IJ18 IJ19, IJ20, IJ21, IJ22 IJ23, IJ24, IJ27, IJ29 IJ30
Reverse spring	The actuator loads a spring. When the actuator is turned off, the spring releases. This can reverse the force/distance curve of the actuator to make it compatible with the force/time requirements of the drop ejection.	<ul style="list-style-type: none"> Better coupling to the ink 	<ul style="list-style-type: none"> Fabrication complexity High stress in the spring 	<ul style="list-style-type: none"> IJ05, IJ11
Actuator stack	A series of thin actuators are stacked. This can be appropriate where actuators require a high electric field strength, such as electrostatic and piezoelectric actuators.	<ul style="list-style-type: none"> Increased travel Reduced drive voltage 	<ul style="list-style-type: none"> Increased fabrication complexity Increased possibility of short circuits due to pinholes 	<ul style="list-style-type: none"> IJ04
Multiple actuators	Multiple smaller actuators are used simultaneously to move the ink. Each actuator need provide only a portion of the force required.	<ul style="list-style-type: none"> Increases the force available from an actuator Multiple actuators can be positioned to control ink flow accurately 	<ul style="list-style-type: none"> Actuator forces may not add linearly, reducing efficiency 	<ul style="list-style-type: none"> IJ12, IJ13, IJ18, IJ20 IJ22, IJ28
Linear Spring	A linear spring is used to transform a motion with small travel and high force into a longer travel, lower force motion.	<ul style="list-style-type: none"> Matches low travel actuator with higher travel requirements Non-contact method of motion transformation 	<ul style="list-style-type: none"> Requires print head area for the spring 	<ul style="list-style-type: none"> IJ15
Coiled actuator	A bend actuator is coiled to provide greater travel in a reduced chip area.	<ul style="list-style-type: none"> Increases travel Reduces chip area Planar implementations are relatively easy to fabricate. 	<ul style="list-style-type: none"> Restricted to planar implementations due to extreme fabrication difficulty in other orientations. 	<ul style="list-style-type: none"> IJ17, IJ21
Flexure region near fixture	A bend actuator has a small region near the fixture point which flexes much more readily than the remainder of the actuator. The actuator flexing is effectively converted from an even coiling to an angular bend, resulting in greater travel of the actuator tip.	<ul style="list-style-type: none"> Simple means of increasing travel of a bend actuator 	<ul style="list-style-type: none"> Care must be taken not to exceed the elastic limit in the flexure area Stress distribution is very uneven Difficult to accurately model with finite element analysis 	<ul style="list-style-type: none"> IJ10, IJ19
Catch	The actuator controls a small catch. The catch either enables or disables movement of an ink pusher which is controlled in a bulk manner.	<ul style="list-style-type: none"> Very low actuator energy Very small actuator size 	<ul style="list-style-type: none"> Complex construction Requires external force Unsuitable for pigmented inks 	<ul style="list-style-type: none"> IJ10

Gears	Gears can be used to increase travel at the expense of duration. Circular gears, rack and pinion, ratchets, and other gearing methods can be used.	<ul style="list-style-type: none"> Low force, low travel actuators can be used Can be fabricated using standard surface MEMS processes 	<ul style="list-style-type: none"> Moving parts are required Several actuator cycles are required More complex drive electronics Complex construction Friction possible Stiction possible Wear possible Unsuitable for pigmented inks 	<ul style="list-style-type: none"> IJ13
Buckle plate	A buckle plate can be used to change a slow actuator into a fast motion. It can also convert a high force, low travel actuator into a high travel, medium force motion.	<ul style="list-style-type: none"> Very fast movement achievable 	<ul style="list-style-type: none"> Must stay within elastic limits of the materials for long device life High stresses involved Generally high power requirement 	<ul style="list-style-type: none"> S. Hirata et al., "An Ink-jet Head Using Diaphragm Microactuator", Proc. IEEE Micro Electro Mechanical Systems, Feb. 1996, pp 418-423. IJ18, IJ27
Tapered magnetic pole	A tapered magnetic pole can increase travel at the expense of force.	<ul style="list-style-type: none"> Linearizes the magnetic force/distance curve 	<ul style="list-style-type: none"> Complex construction 	<ul style="list-style-type: none"> IJ14
Rotary impeller	The actuator is connected to a rotary impeller. A small angular deflection of the actuator results in a rotation of the impeller vanes, which push the ink against stationary vanes and out of the nozzle. The ratio of force to travel of the actuator can be matched to the nozzle requirements by varying the number of impeller vanes.	<ul style="list-style-type: none"> High mechanical advantage 	<ul style="list-style-type: none"> Complex construction Unsuitable for pigmented inks 	<ul style="list-style-type: none"> IJ28

Actuator motion

Actuator motion	Description	Advantages	Disadvantages	Examples
Volume expansion	The volume of the actuator changes, pushing the ink in all directions.	<ul style="list-style-type: none"> ♦ The actuator is relatively non-directional, so placement of the nozzle relative to the actuator is not critical 	<ul style="list-style-type: none"> ♦ High energy is typically required to achieve volume expansion 	<ul style="list-style-type: none"> ♦ Hewlett-Packard Thermal Ink Jet ♦ Canon Bubblejet
Linear, normal to chip surface	The actuator moves in a direction normal to the print head surface. The nozzle is typically in the line of movement.	<ul style="list-style-type: none"> ♦ Efficient coupling to ink drops ejected normal to the surface 	<ul style="list-style-type: none"> ♦ High fabrication complexity may be required to achieve perpendicular motion 	<ul style="list-style-type: none"> ♦ IJ01, IJ02, IJ04, IJ07 ♦ IJ11, IJ14
Linear, parallel to chip surface	The actuator moves parallel to the print head surface. Drop ejection may still be normal to the surface.	<ul style="list-style-type: none"> ♦ Suitable for planar fabrication 	<ul style="list-style-type: none"> ♦ Fabrication complexity ♦ Friction ♦ Stiction 	<ul style="list-style-type: none"> ♦ IJ12, IJ13, IJ15
Membrane push	An actuator with a high force but small area is used to push a stiff membrane which is in contact with the ink.	<ul style="list-style-type: none"> ♦ The effective area of the actuator becomes the membrane area 	<ul style="list-style-type: none"> ♦ Fabrication complexity ♦ Actuator size ♦ Difficulty of integration in a VLSI process 	<ul style="list-style-type: none"> ♦ 1982 Hawkins USP 4,459,601
Rotary	The actuator causes the rotation of some element, such as a grill or impeller	<ul style="list-style-type: none"> ♦ Rotary levers may be used to increase travel ♦ Small chip area requirements 	<ul style="list-style-type: none"> ♦ Device complexity ♦ May have friction at a pivot point 	<ul style="list-style-type: none"> ♦ IJ05, IJ08, IJ13, IJ28
Bend	The actuator bends when energized. This may be due to differential thermal expansion, piezoelectric expansion, magnetostriction, or other form of relative dimensional change.	<ul style="list-style-type: none"> ♦ A very small change in dimensions can be converted to a large motion. 	<ul style="list-style-type: none"> ♦ Requires the actuator to be made from at least two distinct layers, or to have a thermal difference across the actuator 	<ul style="list-style-type: none"> ♦ 1970 Kyser et al USP 3,946,398 ♦ 1973 Stemme USP 3,747,120 ♦ IJ03, IJ06, IJ09, IJ10 ♦ IJ19, IJ23, IJ24, IJ25 ♦ IJ29, IJ30
Straighten	The actuator is normally bent, and straightens when energized.	<ul style="list-style-type: none"> ♦ Can be used with shape memory alloys where the austenitic phase is planar 	<ul style="list-style-type: none"> ♦ Requires careful balance of stresses to ensure that the quiescent bend is accurate 	<ul style="list-style-type: none"> ♦ IJ26
Shear	Energizing the actuator causes a shear motion in the actuator material.	<ul style="list-style-type: none"> ♦ Can increase the effective travel of piezoelectric actuators 	<ul style="list-style-type: none"> ♦ Not readily applicable to other actuator mechanisms 	<ul style="list-style-type: none"> ♦ 1985 Fishbeck USP 4,584,590
Radial constriction	An ink reservoir is squeezed by the actuator, forcing the ink from a constricted nozzle.	<ul style="list-style-type: none"> ♦ Relatively easy to fabricate single nozzles from glass tubing as macroscopic structures 	<ul style="list-style-type: none"> ♦ High force required ♦ Inefficient ♦ Difficult to integrate with VLSI processes 	<ul style="list-style-type: none"> ♦ 1970 Zoltan USP 3,683,212
Coil / uncoil	A coiled actuator uncoils or coils more tightly. The motion of the free end of the actuator ejects the ink.	<ul style="list-style-type: none"> ♦ Easy to fabricate as a planar VLSI process ♦ Small area required, therefore low cost 	<ul style="list-style-type: none"> ♦ Difficult to fabricate for non-planar devices ♦ Poor out-of-plane stiffness 	<ul style="list-style-type: none"> ♦ IJ17, IJ21
Bow	The actuator bows (or buckles) in the middle when energized.	<ul style="list-style-type: none"> ♦ Can increase the speed of travel ♦ Mechanically rigid 	<ul style="list-style-type: none"> ♦ Maximum travel is constrained ♦ High force required 	<ul style="list-style-type: none"> ♦ IJ16, IJ18, IJ27
Push-Pull	A shutter is controlled by two actuators. One actuator pulls the shutter, and the other pushes it.	<ul style="list-style-type: none"> ♦ The structure is pinned at both ends, so has a high out-of-plane rigidity 	<ul style="list-style-type: none"> ♦ Not readily suitable for ink jets which directly push the ink 	<ul style="list-style-type: none"> ♦ IJ18
Curl/ uncurl	A set of actuators curl up to reduce the volume of ink that they enclose.	<ul style="list-style-type: none"> ♦ Good ink flow to the region behind the actuator increases efficiency 	<ul style="list-style-type: none"> ♦ Design complexity 	<ul style="list-style-type: none"> ♦ IJ20
Iris	Multiple vanes enclose a volume of ink. These simultaneously rotate, reducing the volume between the vanes.	<ul style="list-style-type: none"> ♦ Good efficiency 	<ul style="list-style-type: none"> ♦ High fabrication complexity ♦ Not suitable for pigmented inks 	<ul style="list-style-type: none"> ♦ IJ22

Nozzle refill method

Nozzle refill method	Description	Advantages	Disadvantages	Examples
Surface tension	This is the normal way that ink jets are refilled. After the actuator is energized, it typically returns rapidly to its normal position. This rapid return sucks in air through the nozzle opening. The ink surface tension at the nozzle then exerts a small force restoring the meniscus to a minimum area. This force refills the nozzle.	<ul style="list-style-type: none"> ♦ Fabrication simplicity ♦ Operational simplicity 	<ul style="list-style-type: none"> ♦ Low speed ♦ Surface tension force relatively small compared to actuator force ♦ Long refill time usually dominates the total ink jet 	<ul style="list-style-type: none"> ♦ Thermal ink jet ♦ Piezo-electric ink jet ♦ IJ01, IJ02, IJ03, IJ04 ♦ IJ05, IJ05, IJ07, IJ10 ♦ IJ11, IJ12, IJ14, IJ16 ♦ IJ20, IJ22, IJ23, IJ24 ♦ IJ25, IJ26, IJ27, IJ28 ♦ IJ29, IJ30
Shuttered oscillating ink pressure	Ink to the nozzle chamber is provided at a pressure which oscillates at twice the drop ejection frequency. When a drop is to be ejected, the shutter is opened for 3 half cycles: drop ejection, actuator return, and refill. The shutter is then closed to prevent the nozzle chamber emptying during the next negative pressure cycle.	<ul style="list-style-type: none"> ♦ High speed ♦ Low actuator energy, as the actuator need only open or close the shutter, instead of ejecting the ink drop 	<ul style="list-style-type: none"> ♦ Requires common ink pressure oscillator ♦ May not be suitable for pigmented inks 	<ul style="list-style-type: none"> ♦ IJ08, IJ13, IJ15, IJ17 ♦ IJ18, IJ19, IJ21
Refill actuator	After the main actuator has ejected a drop a second (refill) actuator is energized. The refill actuator pushes ink into the nozzle chamber. The refill actuator returns slowly, to prevent its return from emptying the chamber again.	<ul style="list-style-type: none"> ♦ High speed, as the nozzle is actively refilled 	<ul style="list-style-type: none"> ♦ Requires two independent actuators per nozzle 	<ul style="list-style-type: none"> ♦ IJ09
Positive ink pressure	The ink is held a slight positive pressure. After the ink drop is ejected, the nozzle chamber fills quickly as surface tension and ink pressure both operate to refill the nozzle.	<ul style="list-style-type: none"> ♦ High refill rate, therefore a high drop repetition rate is possible 	<ul style="list-style-type: none"> ♦ Surface spill is more likely ♦ Highly hydrophobic print head surfaces are required 	<ul style="list-style-type: none"> ♦ LIFT ♦ Alternative for: ♦ IJ01, IJ02, IJ03, IJ04 ♦ IJ05, IJ05, IJ07, IJ10 ♦ IJ11, IJ12, IJ14, IJ16 ♦ IJ20, IJ22, IJ23, IJ24 ♦ IJ25, IJ26, IJ27, IJ28 ♦ IJ29, IJ30

Nozzle plate construction

Nozzle plate construction	Description	Advantages	Disadvantages	Examples
Electroformed nickel	A nozzle plate is separately fabricated from electroformed nickel, and bonded to the print head chip.	<ul style="list-style-type: none"> ♦ Fabrication simplicity 	<ul style="list-style-type: none"> ♦ High temperatures and pressures are required to bond nozzle plate ♦ Minimum thickness constraints ♦ Differential thermal expansion 	<ul style="list-style-type: none"> ♦ Hewlett Packard Thermal Ink Jet
Laser ablated or drilled polymer	Individual nozzle holes are ablated by an intense UV laser in a nozzle plate, which is typically a polymer such as polyimide or polysulphone	<ul style="list-style-type: none"> ♦ No masks required ♦ Can be quite fast ♦ Some control over nozzle profile is possible ♦ Equipment required is relatively low cost 	<ul style="list-style-type: none"> ♦ Each hole must be individually formed ♦ Special equipment required ♦ Slow where there are many thousands of nozzles per print head ♦ May produce thin burrs at exit holes 	<ul style="list-style-type: none"> ♦ Canon Bubblejet ♦ 1988 Sercei et al., SPIE, Vol. 998 Excimer Beam Applications, pp. 76-83 ♦ 1993 Watanabe et al., USP 5,208,604
Silicon micro-machined	A separate nozzle plate is micromachined from single crystal silicon, and bonded to the print head wafer.	<ul style="list-style-type: none"> ♦ High accuracy is attainable 	<ul style="list-style-type: none"> ♦ Two part construction ♦ High cost ♦ Requires precision alignment ♦ Nozzles may be clogged by adhesive 	<ul style="list-style-type: none"> ♦ K. Bean, IEEE Transactions on Electron Devices, Vol. ED-25, No. 10, 1978, pp 1185-1195 ♦ Xerox 1990 Hawkins et al., USP 4,899,181
Glass capillaries	Fine glass capillaries are drawn from glass tubing. This method has been used for making individual nozzles, but is difficult to use for bulk manufacturing of print heads with thousands of nozzles.	<ul style="list-style-type: none"> ♦ No expensive equipment required ♦ Simple to make single nozzles 	<ul style="list-style-type: none"> ♦ Very small nozzle sizes are difficult to form ♦ Not suited for mass production 	<ul style="list-style-type: none"> ♦ 1970 Zoltan USP 3,683,212
Monolithic, surface micro-machined using VLSI lithographic processes	The nozzle plate is deposited as a layer using standard VLSI deposition techniques. Nozzles are etched in the nozzle plate using VLSI lithography and etching.	<ul style="list-style-type: none"> ♦ High accuracy ($<1 \mu\text{m}$) ♦ Monolithic ♦ Low cost ♦ Existing processes can be used 	<ul style="list-style-type: none"> ♦ Requires sacrificial layer under the nozzle plate to form the nozzle chamber ♦ Surface is fragile to the touch 	<ul style="list-style-type: none"> ♦ LIFT ♦ IJ01, IJ02, IJ04, IJ11 ♦ IJ12, IJ17, IJ18, IJ20 ♦ IJ22, IJ24, IJ27, IJ28 ♦ IJ29, IJ30
Monolithic, etched through substrate	The nozzle plate is a buried etch stop in the wafer. Nozzle chambers are etched in the front of the wafer, and the wafer is thinned from the back side to the etch stop. Nozzles are then etched in the etch stop layer.	<ul style="list-style-type: none"> ♦ High accuracy ($<1 \mu\text{m}$) ♦ Monolithic ♦ Low cost ♦ No differential expansion 	<ul style="list-style-type: none"> ♦ Requires long etch times ♦ Requires a support wafer 	<ul style="list-style-type: none"> ♦ IJ03, IJ05, IJ06, IJ07 ♦ IJ08, IJ09, IJ10, IJ13 ♦ IJ14, IJ15, IJ16, IJ19 ♦ IJ21, IJ23, IJ25, IJ26
No nozzle plate	Various methods have been tried to eliminate the nozzles entirely, to prevent nozzle clogging. These include thermal bubble mechanisms and acoustic lens mechanisms	<ul style="list-style-type: none"> ♦ No nozzles to become clogged 	<ul style="list-style-type: none"> ♦ Difficult to control drop position accurately ♦ Crosstalk problems 	<ul style="list-style-type: none"> ♦ Ricoh 1995 Sekiya et al USP 5,412,413 ♦ 1993 Hadimioglu et al EUP 550,192 ♦ 1993 Elrod et al EUP 572,220
Nozzle slit instead of individual nozzles	The elimination of nozzle holes and replacement by a slit encompassing many actuator positions reduces nozzle clogging, but increases crosstalk due to ink surface waves	<ul style="list-style-type: none"> ♦ No nozzles to become clogged 	<ul style="list-style-type: none"> ♦ Difficult to control drop position accurately ♦ Crosstalk problems 	<ul style="list-style-type: none"> ♦ 1989 Saito et al USP 4,799,068

Drop ejection direction

Ejection direction:	Description	Advantages	Disadvantages	Examples
Edge (‘edge shooter’)	Ink flow is along the surface of the chip, and ink drops are ejected from the chip edge.	<ul style="list-style-type: none"> Simple construction No silicon etching required Good heat sinking via substrate Mechanically strong Ease of chip handling 	<ul style="list-style-type: none"> Nozzles limited to edge High resolution is difficult Fast color printing requires one print head per color 	<ul style="list-style-type: none"> Canon Bubblejet 1979 Endo et al GB patent 2,007,162 Xerox heater-in-pit 1990 Hawkins et al USP 4,899,181
Surface (‘roof shooter’)	Ink flow is along the surface of the chip, and ink drops are ejected from the chip surface, normal to the plane of the chip.	<ul style="list-style-type: none"> No bulk silicon etching required Silicon can make an effective heat sink Mechanical strength 	<ul style="list-style-type: none"> Maximum ink flow is severely restricted 	<ul style="list-style-type: none"> Hewlett-Packard TIJ 1982 Vaught et al USP 4,490,728 IJ02, IJ11, IJ12, IJ20 IJ22
Through chip, forward (‘up shooter’)	Ink flow is through the chip, and ink drops are ejected from the front surface of the chip.	<ul style="list-style-type: none"> High ink flow Suitable for pagewidth print heads High nozzle packing density therefore low manufacturing cost 	<ul style="list-style-type: none"> Requires bulk silicon etching 	<ul style="list-style-type: none"> LIFT IJ04, IJ17, IJ18, IJ24 IJ27, IJ28, IJ29, IJ30
Through chip, reverse (‘down shooter’)	Ink flow is through the chip, and ink drops are ejected from the rear surface of the chip.	<ul style="list-style-type: none"> High ink flow Suitable for pagewidth print heads High nozzle packing density therefore low manufacturing cost 	<ul style="list-style-type: none"> Requires wafer thinning Requires special handling during manufacture 	<ul style="list-style-type: none"> IJ01, IJ03, IJ05, IJ06 IJ07, IJ08, IJ09, IJ10 IJ13, IJ14, IJ15, IJ16 IJ19, IJ21, IJ23, IJ25 IJ26
Through actuator	Ink flow is through the actuator, which is not fabricated as part of the same substrate as the drive transistors.	<ul style="list-style-type: none"> Suitable for piezo-electric print heads 	<ul style="list-style-type: none"> Pagewidth print head requires thousands of connections to drive electronics Cannot be manufactured in standard CMOS fabs Complex assembly required 	<ul style="list-style-type: none"> Epson Stylus Tektronix

Ink type

Ink type	Description	Advantages	Disadvantages	Examples
Aqueous, dye	Water based ink which typically contains: water, dye, surfactant, humectant, and biocide. Modern ink dyes have high water-fastness, light fastness	<ul style="list-style-type: none"> Environmentally friendly No odor 	<ul style="list-style-type: none"> Slow drying Corrosive Bleeds on paper May strikethrough Cockles paper 	<ul style="list-style-type: none"> Most existing ink jets IJ01, IJ02, IJ03, IJ04 IJ05, IJ06, IJ07, IJ08 IJ09, IJ10, IJ11, IJ12 IJ13, IJ14, IJ15, IJ16 IJ17, IJ18, IJ19, IJ20 IJ21, IJ22, IJ23, IJ24 IJ25, IJ26, IJ27, IJ28 IJ29, IJ30
Aqueous, pigment	Water based ink which typically contains: water, pigment, surfactant, humectant, and biocide. Pigments have an advantage in reduced bleed, wicking and strikethrough.	<ul style="list-style-type: none"> Environmentally friendly No odor Reduced bleed Reduced wicking Reduced strikethrough 	<ul style="list-style-type: none"> Slow drying Corrosive Pigment may clog nozzles Pigment may clog actuator mechanisms Cockles paper 	<ul style="list-style-type: none"> IJ02, IJ04, IJ21, IJ26 IJ27, IJ30
Methyl Ethyl Ketone (MEK)	MEK is a highly volatile solvent used for industrial printing on difficult surfaces such as aluminum cans	<ul style="list-style-type: none"> Very fast drying Prints on various substrates such as metals and plastics 	<ul style="list-style-type: none"> Odorous Flammable 	<ul style="list-style-type: none"> IJ01, IJ02, IJ03, IJ04 IJ05, IJ06, IJ07, IJ08 IJ09, IJ10, IJ11, IJ12 IJ13, IJ14, IJ15, IJ16 IJ17, IJ18, IJ19, IJ20 IJ21, IJ22, IJ23, IJ24 IJ25, IJ26, IJ27, IJ28 IJ29, IJ30
Phase change (hot melt)	The ink is solid at room temperature, and is melted in the print head before jetting. Hot melt inks are usually wax based, with a melting point around 80 °C. After jetting the ink freezes almost instantly upon contacting the print medium or a transfer roller.	<ul style="list-style-type: none"> No drying time- ink instantly freezes on the print medium Almost any print medium can be used No paper cockle occurs No wicking occurs No bleed occurs No strikethrough occurs 	<ul style="list-style-type: none"> High viscosity Printed ink typically has a 'waxy' feel Printed pages may 'block' Ink temperature may be above the curie point of permanent magnets Ink heaters consume power Long warm-up time 	<ul style="list-style-type: none"> Tektronix 1989 Nowak USP 4,820,346 IJ01, IJ02, IJ03, IJ04 IJ05, IJ06, IJ08, IJ09 IJ11, IJ12, IJ13, IJ14 IJ15, IJ16, IJ17, IJ18 IJ19, IJ20, IJ21, IJ22 IJ23, IJ24, IJ25, IJ26 IJ27, IJ28, IJ29, IJ30
Oil	Oil based inks are extensively used in offset printing. They have advantages in improved characteristics on paper (especially no wicking or cockle). Oil soluble dyes and pigments are required.	<ul style="list-style-type: none"> High solubility medium for some dyes Does not cockle paper Does not wick through paper 	<ul style="list-style-type: none"> High viscosity: this is a significant limitation for use in ink jets, which usually require a low viscosity. Some short chain and multi-branched oils have a sufficiently low viscosity. Slow drying 	<ul style="list-style-type: none"> IJ01, IJ02, IJ03, IJ04 IJ05, IJ06, IJ07, IJ08 IJ09, IJ10, IJ11, IJ12 IJ13, IJ14, IJ15, IJ16 IJ17, IJ18, IJ19, IJ20 IJ21, IJ22, IJ23, IJ24 IJ25, IJ26, IJ27, IJ28 IJ29, IJ30
Microemulsion	A microemulsion is a stable, self forming emulsion of oil, water, and surfactant. The characteristic drop size is less than 100 nm. and is determined by the preferred curvature of the surfactant.	<ul style="list-style-type: none"> Stops ink bleed High dye solubility Water, oil, and amphiphilic soluble dyes can be used Can stabilize pigment suspensions 	<ul style="list-style-type: none"> Viscosity higher than water Cost is slightly higher than water based ink High surfactant concentration required (around 5%) 	<ul style="list-style-type: none"> IJ01, IJ02, IJ03, IJ04 IJ05, IJ06, IJ07, IJ08 IJ09, IJ10, IJ11, IJ12 IJ13, IJ14, IJ15, IJ16 IJ17, IJ18, IJ19, IJ20 IJ21, IJ22, IJ23, IJ24 IJ25, IJ26, IJ27, IJ28 IJ29, IJ30

APPENDIX C – IJ SERIES INK JETS

This appendix sets out a series of novel arrangements of ink jets constructed in accordance with the physical variables as set out in appendix B. Each of these ink jets include features that can be combined with other inkjets in accordance with requirements.

IJ01 DIRECT PLUNGER ELECTROMAGNETIC INK JET

Ink Jet IJ01 is described with reference to the accompanying drawings in which:

Fig. C01.1 illustrates a cross section of a single nozzle and actuator of a 1200 dpi print head, shown firing an ink drop.

Fig. C01.2 illustrates an exploded view of the nozzle.

IJ01 is a direct firing electromagnetic ink jet, where an electromagnet attracts a soft magnetic plunger which ejects the ink. When data signals distributed on the print head indicate that a particular nozzle is to eject a drop of ink, the drive transistor for that nozzle is turned on. This energizes a solenoid, which induces a magnetic field in a nickel iron fixed plate and movable plunger. When the solenoid is energized, the plunger is attracted to the fixed plate across a gap. The plunger pushes against the ink, creating a high pressure in the nozzle chamber, causing ink to be squirted out of the nozzle. Ink trapped between the plunger and the solenoid is squirted out of the holes in the top of the plunger. This prevents trapped ink increasing the pressure on the plunger, and thereby requiring more magnetic force to move the plunger. After approximately 2 μ s, the current to the solenoid is turned off. At the same time, or at a slightly later time, a reverse current is applied, approximately half of the forward current. As the plunger will carry some residual magnetism, this causes the plunger to move backwards towards its nominal position. A spring also helps to return the plunger. The reverse current is turned off before the magnetization of the plunger is reversed, which would cause the plunger be attracted to the fixed plate again. The return of the plunger to its quiescent position causes a low pressure in the ink chamber. This causes ink to begin flowing from the ejected drop back into the nozzle, and also ingests air into the chamber. The forward velocity of the drop and backward velocity of the ink in the chamber are resolved by the ink drop breaking off from the ink in the nozzle. The ink drop then continues to travel towards the recording medium under its own momentum. The nozzle refills due to the surface tension of the ink at the nozzle tip. Shortly after the time of drop breakoff, the meniscus at the nozzle tip is approximately a concave hemisphere. The surface tension exerts a net forward force on the ink, which results in the nozzle refilling. The repetition rate of the ink jet is principally determined by the nozzle refill time, which will be approximately 100 μ s, depending upon device geometry, ink surface tension, and the volume of the ejected drop.

In an alternative arrangement ink in the region of the solenoid is also squirted out of the nozzle, through a series of posts that complete the magnetic circuit. The ink around the solenoid has a higher fluidic resistance before reaching the nozzle than ink directly between the nozzle and the plunger. It will therefore exert a greater pressure on the plunger than if the ink were allowed to escape, requiring greater magnetic force to move the plunger. However, this ink adds to the quantity of ink which is squirted out of the nozzle, thereby allowing a smaller plunger travel, and a smaller gap between the plunger and the fixed magnetic plate. This smaller gap means that substantially more magnetic force is available to move the plunger.

The drop firing rate is around 7 KHz. The ink jet head is suitable for fabrication as a monolithic pagewide print head. The print head has a 'down shooter' configuration.

IJ02 ELECTROSTATIC INK JET

Ink Jet IJ02 is described with reference to the accompanying drawings in which:

Fig. C02.1 illustrates a single nozzle and actuator of a 1600 dpi print head. The central hole is the nozzle, with a 8 μm radius. The smaller holes are to allow etching of the sacrificial layers during fabrication. Ink flows across the chip surface, and enters the nozzle chamber via ports at the periphery.

Fig. C02.2 illustrates a cross section detail showing the nozzle wall, concertina ring, electrostatic plates, and air vents.

IJ02 is a reverse firing electrostatic ink jet. To prepare for firing a drop, a voltage difference is applied to a pair of parallel conductive plates below the nozzle chamber. These conductive plates are large relative to the nozzle, and are separated by a small air gap. The applied voltage induces charges on the plates proportional to the capacitance of the two plates. The opposite charges result in an attractive electrostatic force, which causes the upper plate to move towards the lower plate. The upper plate is suspended on a concertina spring, so movement is achieved by bending the spring, not stretching the upper plate. The lower plate is embedded in the substrate material. The air between the two plates is vented to the outside, via air vents along the periphery of the nozzle chambers. The gap between the two plates is lined with a hydrophobic material, such as polytetrafluoroethylene (PTFE), which prevents any spilled ink from entering the air gap and filling it by capillarity. The PTFE lining also prevents stiction after sacrificial etching.

The voltage applied to the plates is maintained at least long enough for the nozzle chamber to refill, and replace the ink displaced by the movement of the upper plate. The nozzle refill is achieved by the surface tension of the ink at the nozzle, and requires approximately 100 μs , depending upon device geometry, ink surface tension, and the volume of the ejected drop.

To fire the drop, the voltage difference between the plates is removed. The elastic forces in the concertina spring cause the upper plate to move back to its normal position, displacing ink as it moves. The displaced ink is ejected from the nozzle.

The voltage is reapplied to the plates after about 5 μs , and maintained until the next drop is to be fired. The re-application of the voltage causes the upper plate to move down again while the ink drop is being ejected. This helps with the drop break-off process. The exact timing of the re-application of the voltage can be altered to minimize satellite drop formation.

The forward velocity of the drop and backward velocity of the upper electrostatic plate are resolved by the ink drop breaking off from the ink in the nozzle. The ink drop then continues to travel towards the recording medium under its own momentum. The repetition rate of the ink jet is principally determined by the nozzle refill time.

The drop firing rate is around 7 KHz. While relatively large, the ink jet head can be fabricated as a monolithic pagewide print head. The print head has a 'roof shooter' configuration.

IJ03 PLANAR THERMOELASTIC BEND ACTUATOR INK JET

Ink Jet IJ03 is described with reference to the accompanying drawings in which:

Fig. C03.1 illustrates a view of a single 1600 dpi nozzle actuator from the inside of the ink reservoir. The top layer is ITO, the layer below that is aluminum, followed silicon nitride.

Fig. C03.2 illustrates a cross section of the nozzle and actuator. The bottom layer is boron doped silicon etch stop. The crystallographically etched nozzle chamber is shown in the silicon layer. The nozzle radius is 8 μm for 1600 dpi operation.

IJ03 has one thermoelastic bend actuator for each nozzle. The bend actuator is a planar layer of two materials separated by a small gap. It is mechanically fixed to the substrate at one end, and the other end is free to move.

The two materials comprising the actuator are electrically joined at the end which is mechanically free. The mechanically fixed end also provides the electrical connections between the drive circuitry and the two plates.

When energized, the actuator bends into the substrate towards a nozzle, pushing on ink in the nozzle chamber. This ink is pushed out of the nozzle, forming the ink drop. As the actuator is cooled by the ink, it returns to its nominal position, withdrawing ink from the nozzle and aiding in drop separation.

The actuator is composed of 4 layers:

- 1) A upper layer of a high resistivity material such as indium-tin oxide (ITO). This layer dissipates the vast majority of the electrical energy passed through the actuator as heat. The resultant temperature rise causes this layer to expand. The ideal characteristics of this layer include a high resistivity, a high Young's modulus, and a high coefficient of thermal expansion.
- 2) A gap. This gap maintains electrical and mechanical separation between the two conductive layers.
- 3) A lower layer of a high conductivity material, such as aluminium. As the resistance of this layer is very small, a negligible amount of heat is dissipated. The layer's characteristics should include a low resistivity, a low Young's modulus, and a low coefficient of thermal expansion.
- 4) A stiffening plate of silicon nitride approximately 80% of the length of the actuator is attached at the free end. This stiffener prevents the lower layer from bending evenly, forcing most of the bend to occur near the fixed end of the actuator. In this manner, a greater deflection of the actuator tip is achieved.

The print head is fabricated as a CMOS device fabricated on a wafer with an epitaxial layer of high concentration boron doped silicon, and a subsequent 10 μm thick layer of lightly doped epitaxial silicon. After CMOS processing (in the lightly doped epitaxial silicon) is complete, MEMS post processing is performed. This includes a nitride passivation layer, an aluminum layer, a sacrificial oxide layer, and an ITO layer. The sacrificial oxide is etched, followed by a crystallographic wet etch of silicon which forms the nozzle chamber as a pit in the front surface silicon. The wafer is then bonded to an ink channel wafer, and the entire back side of the wafer is etched until the boron doped layer is reached. The nozzle holes are then masked and plasma etched.

Not shown are the layers of passivation material required to prevent corrosion of the aluminum and ITO layers by the ink.

The drop firing rate is around 7 KHz. The ink jet head is suitable for fabrication as a monolithic pagewide print head. The print head has an 'roof shooter' configuration.

IJ04 STACKED ELECTROSTATIC/PIEZOELECTRIC/ELECTROSTRICTIVE INK JET

Ink Jet IJ04 is described with reference to the accompanying drawings in which:

Fig. C04.1 illustrates a view of a single 1600 dpi nozzle and actuator stack. The actuator stack consists of 40 layers of elastomer sandwiched between electrodes.

Fig. C04.2 illustrates a cross section detail showing one end of the actuator stack, next to the silicon nitride chamber filter.

IJ04 uses a stack of electrostatic plates separated by an elastomer dielectric. Alternate plates are connected to one electrode, and the other plates are connected to another electrode. When a voltage is applied across the electrodes, alternate charges on alternate plates cause the plates to attract each other, compressing the elastomer. For efficient operation, manufacturability, and reliability of the actuator, the properties of the elastomer are critical. To achieve a small actuator size, the elastomer should have a very low bulk modulus, a high dielectric constant, and a high dielectric strength. For high manufacturability, it must be able to be applied in thin (approx. 0.2 μm) pin-hole free layers by spin coating, evaporation, or other means. It must also survive subsequent processing steps which may occur at elevated temperatures. For reliability, the elastomer should have high stability, and be chemically inert.

Materials such as styrene ethylene butylene styrene block copolymer (trade name C-Flex R70-190 from Consolidated) have a very low 100% modulus of 0.14 MPa, which is not much higher than air at

atmospheric pressure. However, this material is not ideal in all respects, as the dielectric constant and dielectric strength are in the normal polymeric range.

The voltage applied to the alternate plates is maintained at least long enough for the nozzle chamber to refill, and replace the ink displaced by the compression of the stack. The nozzle refill is achieved by the surface tension of the ink at the nozzle, and requires approximately 100 μ s, depending upon device geometry, ink surface tension, and the volume of the ejected drop. To fire the drop, the voltage difference between the plates is removed. The elastic forces in the elastomer cause the stack to return to its normal volume, ejecting ink from the nozzle.

The voltage is reapplied to the plates after about 5 μ s, and maintained until the next drop is to be fired. The re-application of the voltage causes the stack to compress again while the ink drop is being ejected. This helps with the drop break-off process. The exact timing of the re-application of the voltage can be altered to minimize satellite drop formation.

In an alternative version, thin films which respond mechanically to the electrostatic field of the electrodes can be used in place of the elastomer. Such materials include:

- 1) Piezoelectric materials such as PZT
- 2) Electrostrictive materials such as PLZT
- 3) Electrically switched transitions between ferroelectric (FE) to antiferroelectric (AFE) phases of a material such as PLZSnT.

The area of the stack is determined by the strain that the material achieves under the applied voltage. Piezoelectric and electrostrictive materials require a larger stack area due to the small strains achieved. With these materials, the stack can be made to expand as well as contract. With an expanding stack, the voltage is applied when the drop is to be fired. A stack which both expands and contracts (such as piezoelectric materials) can be driven with a bipolar voltage to achieve twice the peak-to-peak strain.

IJ05 REVERSE SPRING LEVER ELECTROMAGNETIC INK JET

Ink Jet IJ05 is described with reference to the accompanying drawings in which:

Fig. C05.1 illustrates a single nozzle, plunger, spring and actuator of a 1600 dpi print head, viewed from inside the ink reservoir.

Fig. C05.2 illustrates an exploded view of a nozzle, plunger, spring and actuator.

IJ05 is an ink jet print head which has one magnetic actuator for each nozzle. Instead of directly ejecting a drop when the actuator solenoid is energized, the drop is ejected when the solenoid is turned off. The magnetic actuator is used to load a spring connected to a plunger. It is the return of the spring to its non-stressed position which moves the plunger and ejects the drop.

The advantages of this are that the plunger velocity is much more constant over the duration of the drop ejection stroke, and that the piston or plunger can be entirely removed from the ink chamber during the ink fill stage, reducing nozzle refill time.

When data signals distributed on the print head indicate that a particular nozzle is to eject a drop of ink, the drive transistor for that nozzle is turned on. This energizes a solenoid, which induces a magnetic field in fixed plate and a movable plunger, constructed from a soft magnetic material such as nickel iron (NiFe). The solenoid power is turned to the maximum current for long enough to move the plunger to its stop position (approximately 2 μ s). The piston is withdrawn from the nozzle chamber, drawing air into the chamber through the nozzle.

The solenoid current is turned to a 'keeper' level while the nozzle fills. The keeper power level is sufficient to maintain the movable pole against the stop. This will typically be substantially less than the maximum current level as the magnetic gap is at a minimum. During the 'keeper' phase, the meniscus at the nozzle tip is approximately a concave hemisphere. The surface tension exerts a net force on the ink, which results in the nozzle refilling, replacing the volume of the piston withdrawal with ink. This process takes approximately 100 μ s.

The solenoid current is then reversed, at around half of the maximum current. The reversal is to demagnetize the magnetic plate and plunger, and to initiate the return of the piston to its nominal

position. The piston is accelerated to its nominal position both by the magnetic repulsion, and by the stressed spring. The force on the piston is greatest at the beginning of the stroke, and slows as the spring elastic stress falls to zero. As a result, the acceleration is high at the beginning of the stroke, and the ink velocity is much more uniform during the stroke. This results in an increased operating tolerance before ink flow over the print head front surface can occur. When the residual magnetism of the plunger is at a minimum, the solenoid reverse current is turned off. The piston continues to move towards the quiescent position. The piston overshoots due to its inertia. Overshoot in the piston movement achieves two things: greater ejected drop volume and velocity, and improved drop breakoff as the piston returns from overshoot to its quiescent position.

The piston return is caused by the spring, which is now stressed in the opposite direction. This motion 'sucks' some of the ink back into the nozzle, causing the ink ligament connecting the ink drop to the ink in the nozzle to thin. The forward velocity of the drop and backward velocity of the ink in the chamber are resolved by the ink drop breaking off from the ink in the nozzle. The piston is then at the quiescent position until the next drop ejection cycle.

The drop firing rate is around 7 KHz. The print head has a 'down shooter' configuration.

IJ06 LORENZ PADDLE ELECTROMAGNETIC INK JET

Ink Jet IJ06 is described with reference to the accompanying drawings in which:

Fig. C06.1 illustrates a view of a single nozzle and actuator of a 1600 dpi print head seen from the inside of the ink reservoir. The paddle consists of a two layer copper coil which is embedded in silicon nitride. The print head is placed in a strong (1 Tesla) external magnetic field which is parallel to the plane of the wafer, and in the left-right plane of the paper as shown here.

Fig. C06.2 illustrates a cross section of the ink jet nozzle and actuator, showing the paddle deflected by the Lorentz force acting on the energized solenoid, which is in an external magnetic field.

IJ06 is a direct firing electromagnetic ink jet. Each nozzle has an associated ink chamber, which is etched into the substrate. On the opposite side of the chamber is a paddle which contains a solenoid. The entire print head is in a constant magnetic field, which may be provided by a permanent magnet, configuration of permanent magnets, or an electromagnet. For optimum efficiency, the constant magnetic field should be as strong as practical. Magnetic fields slightly greater than 1 Tesla can be achieved with neodymium iron boron (NdFeB) permanent magnets.

When data signals distributed on the print head indicate that a particular nozzle is to eject a drop of ink, the drive transistor for that nozzle is turned on. This energizes the solenoid. The solenoid current interacts with the fixed magnetic field, producing a torque on the paddle. One side of the paddle moves towards the nozzle, pushing ink out of the nozzle. The other side of the paddle has a force directed away from the nozzle. The paddle is asymmetric, with the side that moves towards the nozzle being longer in relation to the effective fulcrum, thereby providing more torque.

When the solenoid is energized, the paddle pushes against the ink, creating a high pressure in the nozzle chamber. This high pressure causes ink to be squirted out of the nozzle. After approximately 2 μ s, the current to the solenoid is turned off. The springs connecting the paddle to the substrate quickly return of the paddle to its quiescent position. This causes ink to begin flowing from the ejected drop back into the nozzle, and also ingests air into the chamber. The forward velocity of the drop and backward velocity of the ink in the chamber are resolved by the ink drop breaking off from the ink in the nozzle. The ink drop then continues to travel towards the recording medium under its own momentum. The nozzle refills due to the surface tension of the ink at the nozzle tip. The repetition rate of the ink jet is principally determined by the nozzle refill time, which will be approximately 100 μ s, depending upon device geometry, ink surface tension, and the volume of the ejected drop.

Two springs provide mechanical connection of the paddle to the substrate. A copper wire embedded in each spring electrically connects the solenoid to the drive circuitry.

The drop firing rate is around 7 KHz. The ink jet head is suitable for fabrication as a monolithic page-wide print head. The print head has a 'down shooter' configuration.

IJ07 PERMANENT MAGNET ELECTROMAGNETIC INK JET

Ink Jet IJ07 is described with reference to the accompanying drawings in which:

Fig. C07.1 illustrates a cross section of a single nozzle and actuator of a 1600 dpi print head after the permanent magnet plunger has returned to its quiescent position after firing a drop.

Fig. C07.2 illustrates an exploded view of a single nozzle and actuator.

IJ07 is a direct firing ink jet print head. For each nozzle there is a solenoid and a permanent magnet piston. For maximum efficiency, a rare earth magnet such as neodymium iron boron (NdFeB) is used. As thousands of magnetic pistons are required in a pagewidth print head, they are fabricated using VLSI thin film deposition and etching techniques. The pistons are magnetized by placing the completed wafer in an intense magnetic field. When the solenoid is energized, it attracts the magnetic piston, causing ink to be ejected from the nozzle. After approximately 2 μ s, the current to the solenoid is turned off. A spring attached to the piston returns it to its nominal quiescent position. This causes the ink drop to break off from the ink in the nozzle, and move towards the print medium. The nozzle refills due to the surface tension of the ink at the nozzle tip. In the quiescent position the piston is fully removed from the nozzle chamber, allowing faster refill.

The print head is fabricated as a CMOS device fabricated on a wafer with an epitaxial layer of heavily boron doped silicon, and a subsequent 10 μ m thick layer of lightly doped epitaxial silicon. After CMOS processing (in the lightly doped epitaxial silicon) is complete, MEMS post processing is performed. This includes a nitride passivation layer, a copper coil layer, a sacrificial oxide layer, a second nitride passivation layer, a NdFeB layer, and a third nitride layer which provides both passivation and the spring. The sacrificial oxide is etched, followed by a crystallographic wet etch of silicon which forms the nozzle chamber as a pit in the front surface silicon. The wafer is then bonded to an ink channel wafer, and the entire back side of the wafer is etched until the boron doped layer is reached. The nozzle holes are then masked and plasma etched in the boron doped layer.

The drop firing rate is around 7 KHz. The ink jet head is suitable for fabrication as a monolithic pagewidth print head. The print head has a 'down shooter' configuration.

IJ08 PLANAR SWING GRILL ELECTROMAGNETIC INK JET

Ink Jet IJ08 is described with reference to the accompanying drawings in which:

Fig. C08.1 illustrates a single nozzle and actuator of a 1600 dpi ink jet viewed from the inside of the ink reservoir. The nozzle actuator is in the quiescent position, with the shutter grill blocking ink access to the nozzle chamber.

Fig. C08.2 illustrates an exploded view of the nozzle.

IJ08 is a magnetically actuated ink jet which uses a pivoting lever to increase the travel of a shutter grill.

An oscillating ink pressure is used to eject ink from the nozzles. Each nozzle has an associated shutter grill, which normally blocks the slots in a fixed grill over the nozzle chamber. The shutter grill is moved to avoid blocking the fixed grill slots by the electromagnetic actuator (a solenoid) whenever an ink drop is to be fired.

The solenoid attracts a soft magnetic (NiFe) bar. The bar is attracted by the solenoid on both sides of a central point, about which it rotates approximately 4 degrees. The bar is attached to the shutter grill, which rotates with the bar, exposing slots in the fixed grill. A one micron movement of the bar ends towards the solenoid results in an approximately 6 micron movement at the outside rim of the shutter grill. This dramatically improves the efficiency of the system, as the magnetic field falls off strongly with distance.

The surface of the wafer is directly immersed in the ink reservoir, or in relatively large ink channels. An ultrasonic transducer (for example, a piezoelectric transducer) is positioned in the reservoir. The transducer oscillates the ink pressure at approximately 100 KHz. The ink pressure oscillation is sufficient that ink drops would be ejected from the nozzle were it not blocked by the shutter.

When data signals distributed on the print head indicate that a particular nozzle is to eject a drop of ink, the drive transistor for that nozzle is turned on. This energizes the actuator, which moves the shutter so that it is not blocking the ink chamber. The peak of the ink pressure variation causes the ink to be squirted out of the nozzle. As the ink pressure goes negative, ink is drawn back into the nozzle, causing drop break-off. The shutter is kept open until the nozzle is refilled on the next positive pressure cycle. It is then shut to prevent the ink from being withdrawn from the nozzle on the next negative pressure cycle.

Each drop ejection takes two ink pressure cycles, resulting in a 50 KHz drop firing rate.

The amplitude of the ultrasonic transducer can be altered in response to the viscosity of the ink (which is typically affected by temperature), and the number of drops which are to be ejected in the current cycle. This amplitude adjustment can be used to maintain consistent drop size in varying environmental conditions.

The nozzle chamber is formed using an anisotropic crystallographic etch of the silicon substrate. Etchant access to the substrate is via the slots in the fixed grill. The device is manufactured on <100> silicon with a buried boron etch stop layer.

IJ09 PUMP ACTION REFILL INK JET

Ink Jet IJ09 is described with reference to the accompanying drawings in which:

Fig. C09.1 illustrates a single nozzle and actuator of a 1600 dpi print head, showing the two thermal bend actuators. The view is from the inside of the ink reservoir.

Fig. C09.2 illustrates a cross section of quiescent position.

Fig. C09.3 illustrates a drop firing actuator (A₁) energized.

Fig. C09.4 illustrates a drop break-off and bubble ingestion.

Fig. C09.5 illustrates a refill actuator (A₂) energized.

Fig. C09.6 illustrates a chamber full during slow return of A₂.

Fig. C09.7 illustrates the combined activation of A₁ and A₂.

IJ09 has two thermoelastic bend actuator for each nozzle. One actuator (A₁) fires the drop, and the other actuator (A₂) refills the nozzle. With this method, a high drop repetition rate of approximately 50 KHz can be obtained. This is because there is no need to wait around 100 μ s for surface tension to refill the nozzle after each drop is fired.

When a drop is to be fired, A₁ is energized by passing a current through an electrothermal heater embedded near the top surface of the actuator. The actuator is made of polytetrafluoroethylene (PTFE). PTFE has a very high coefficient of thermal expansion (approximately 770×10^{-6} , or around 380 times that of silicon). The top region of the actuator is heated, but the bottom region is not. Differential thermal expansion causes the actuator to bend downwards, pushing on the ink in the ink chamber and expelling a drop from the nozzle.

When the heater current is turned off, the paddle begins to return to its quiescent position. The paddle return 'sucks' some of the ink back into the nozzle, causing the ink drop to break off from the ink in the nozzle. At the same time, air is ingested into the nozzle chamber through the nozzle. At this time, A₂ is actuated, causing the nozzle to refill. The power to A₂ is released slowly, so that the return of A₂ to its quiescent position matches the rate of refill of the ink chamber by surface tension. In this manner, the nozzle chamber stays approximately full during the relatively long refill period. High printing speeds are achieved by allowing the next drop to be fired at any time during the slow return of A₂. This is done by firing A₁ again. A new drop can be fired shortly after the previous drop, as the nozzle chamber remains full after A₂ is fired. A₂ is released approximately 1 μ s after A₁ is fired the second time. The nozzle is refilled a second time by firing A₂ again after A₁ returns to its quiescent position. The exact timing, energy, and duration of the pulses to A₁ and A₂ are controlled to compensate for the minor drop size differences resulting from firing A₁ at different times during the return of A₂.

The ink jet head is suitable for fabrication as a monolithic pagewide print head. The print head has an 'down shooter' configuration. The PTFE surfaces are rendered hydrophilic by high energy plasma bombardment in an ammonia atmosphere. Alternatively, a hydrophilic polymer can be used, albeit with somewhat lower efficiency.

IJ10 PULSED MAGNETIC FIELD INK JET

Ink Jet IJ10 is described with reference to the accompanying drawings in which:

Fig. C10.1 illustrates a cross section of a single nozzle and actuator of a 1600 dpi print head, showing the passive magnetic paddle, and the thermal bend actuator catch. The catch is actuated, preventing the paddle from moving in response to the pulsed magnetic field. The silicon nitride spring is obscured behind the paddle.

Fig. C10.2 illustrates a cross section showing the catch turned off, allowing the paddle to be moved by the pulsed magnetic field, thereby expelling a drop.

IJ10 utilizes an external pulsed magnetic field to attract a spring loaded paddle which pushes ink out of the nozzle. The pulsed magnetic field acts upon all of the paddles (one per nozzle) in the print head simultaneously. When unconstrained, each paddle fires a drop of ink for each pulse of the magnetic field. The magnetic field is pulsed when any drop is to be fired.

For those nozzles which are not to fire a drop, the catch is actuated, preventing the paddle from moving in response to the magnetic pulse.

The catch is actuated by a small electrothermal bend actuator. The forces required are only a few μN , and the displacement required is only a few μm . Therefore, the energy required to actuate the catch is very small. The energy required to eject the ink drops is provided by the external pulsed electromagnet. The magnetic material in the paddle can be either a soft ferromagnetic material such as nickel iron (NiFe), or a permanent magnetic material such as neodymium iron boron (NdFeB). If a permanent magnetic material is used, the material should be magnetized during wafer processing, after the last high temperature processing step.

Unlike most other ink jet types, the actuator is turned on when a drop is not to be fired, rather than when it is to be fired. This means that the number of drops fired is not proportional to the energy dissipated in the actuators. As a result, self-cooling by the ejected ink drops cannot be relied upon. Fortunately, the power consumption of the thermal actuators is so low that power dissipation is not a significant problem with this type of print head.

After a drop is fired, the nozzle refills due to the surface tension of the ink at the nozzle tip. The repetition rate of the ink jet is principally determined by the nozzle refill time, which will be approximately 100 μs , depending upon device geometry, ink surface tension, and the volume of the ejected drop.

The drop firing rate is around 7 KHz. The ink jet head is suitable for fabrication as a monolithic pagewide print head. The print head has a 'down shooter' configuration.

IJ11 TWO COIL REVERSE FIRING ELECTROMAGNETIC INK JET

Ink Jet IJ11 is described with reference to the accompanying drawings in which:

Fig. C11.1 illustrates a single nozzle, static coil, and moving coil of a 1600 dpi print head.

Fig. C11.2 illustrates an exploded view of a nozzle, moving coil, and static coil. The nozzle chamber is fabricated from silicon nitride using an oxide sacrificial layer. Ink flow into the chamber is via slots in the rim of the chamber.

IJ11 is an ink jet print head which has a static coil and a movable coil for each nozzle. When energized, the static and movable coils are attracted towards each other, 'loading' a spring. The drop is ejected from the nozzle when the coils are de-energized. The movable coil forms a plunger, and is rapidly returned to its quiescent position by the loaded spring.

The use of coils without soft magnetic cores simplifies fabrication of the device, as no unusual materials such as NiFe are required. Also, there is no magnetic material which needs to be demagnetized. The disadvantage of a coil arrangement without soft magnetic materials is low magnetic flux density, requiring higher currents to achieve equivalent forces.

The coil metal is preferably copper, for high conductivity and increased resistance to electromigration. The copper coils are encased in silicon nitride for corrosion resistance and mechanical support.

When data signals distributed on the print head indicate that a particular nozzle is to eject a drop of ink, the drive transistor for that nozzle is turned on. This energizes both the static coil and the moving coil. The coils are energized for long enough for the moving coil to reach its stop position (approximately $2\mu\text{s}$).

The coil current is turned to a 'keeper' level while the nozzle refills. The keeper power level is substantially less than the maximum current level because the magnetic gap is at a minimum when the moving coil is at the 'stop' position. During the 'keeper' phase, the meniscus at the nozzle tip is approximately a concave hemisphere. The surface tension exerts a net force on the ink, which results in the nozzle refilling, replacing the volume of the moving coil withdrawal with ink. This process takes approximately $100\mu\text{s}$.

The coil current is then turned off. The plunger is accelerated to its nominal position by the stressed spring. The force on the plunger is greatest at the beginning of the stroke, and slows as the spring elastic stress falls to zero. As a result, the acceleration is high at the beginning of the stroke, and the ink velocity is much more uniform during the stroke. The moving coil is then at the quiescent position until the next drop ejection cycle.

The drop firing rate is around 7 KHz. The ink jet head is suitable for fabrication as a monolithic pagewide print head. The print head has a 'roof shooter' configuration.

IJ12 VARIABLE RELUCTANCE LINEAR STEPPER ACTUATOR INK JET

Ink Jet IJ12 is described with reference to the accompanying drawings in which:

Fig. C12.1 illustrates a single nozzle and actuator of a 1600 dpi ink jet. The moving pole is sandwiched between 12 coils which are driven in three phases. The order of the three phases determines whether the moving pole pushes the piston into the nozzle chamber (at rear) or withdraws it from the chamber.

Fig. C12.2 illustrates an exploded view of the variable reluctance linear stepper actuator, the piston, the nozzle chamber, and the nozzle. The nozzle radius is $8\mu\text{m}$ for 1600 dpi half tone operation. Ink ingress to the nozzle chamber is via the integrated filter in the wall of the nozzle chamber.

IJ12 is a an ink jet head utilizing one miniature linear magnetic stepper actuator for each nozzle. The actuators are bulk fabricated on the wafer surface using VLSI processing techniques.

The general class of linear magnetic actuators include the Linear Induction Actuator (LIA), Linear Permanent Magnet Synchronous Actuator (LPMSA), Linear Reluctance Synchronous Actuator (LRSA), Linear Switched Reluctance Actuator (LSRA), and the Linear Stepper Actuator (LSA). All of these types of actuator can be adapted to operate as an ink jet actuator. The LSA is particularly suitable as it is driven by digital signals rather than phased sinusoidal signals.

The LSA is a relatively complex way of fabricating an ink jet actuator, and is also not as area or power efficient as some other types of actuator. However, it does have the advantage that the travel of the piston can be accurately controlled at multiple points. This makes it particularly suitable for contone operation.

In the example shown, the LSA is a double sided flat variable reluctance linear stepper actuator. It contains twelve solenoids, wired in series in three separate phases. The central movable core of soft magnetic material (e.g. NiFe) has a number of 'teeth' whose spacing is harmonically related to the spacing of the solenoid poles. These teeth result in a variable reluctance in the magnetic circuit. When the three sets of solenoids are sequentially energized in the order 1,2,3,1,2,3 the moving pole is 'stepped' towards the nozzle chamber, where it pushes a piston into the chamber. As the piston moves into the chamber, it ejects an ink drop. The piston is withdrawn from the chamber by energizing the solenoids in the order 3,2,1,3,2,1. The rate of movement of the piston in both directions can be finely controlled by varying the times that the solenoids are energized.

The piston is hydrophobic (e.g. PTFE), so ink does not flow out of the nozzle by capillarity between the piston and the nozzle chamber wall. Due to the hydrophobicity, the piston becomes suspended

in the middle of the ink meniscus, eliminating mechanical friction with the side walls of the nozzle chamber.

The drop firing rate is around 7 KHz. This is limited by the nozzle refill time. Nozzle refill requires approximately 100 μ s, as it occurs as a result of surface tension. The print head has a 'roof shooter' configuration.

IJ13 GEAR DRIVEN SHUTTER INK JET

Ink Jet IJ13 is described with reference to the accompanying drawings in which:

Fig. C13.1 illustrates a single nozzle and actuators of a 1600 dpi print head viewed from the inside of the ink reservoir. The ink pressure in the reservoir oscillates with sufficient amplitude to eject drops when the shutter is open. The actuators are in a external magnetic field, which is normal to the plane of the device. Opposing pairs of actuators are pulsed 144 times to turn the gears and open or close the shutter.

Fig. C13.2 illustrates a cross section of the nozzle, showing the gear construction.

IJ13 uses gears to allow a high speed actuator which produces a tiny force and a 1 micron travel to move a shutter across a nozzle chamber.

In the example shown, two actuators are each pulsed 144 times to move the shutter from an open to a closed condition. A second pair of actuators are each pulsed 144 times to move the shutter back.

The four actuators are each a single turn of wire which are immersed in a constant magnetic field, with the field being normal to the plane of the actuators. When a current is passed through each wire, the Lorentz force acting on the wire bends it a short distance. The wire incorporates two concertina springs, so the wire bends rather than stretches.

The gears are fabricated using standard MEMS processes, and may be made from two layers of polysilicon separated by two layers of sacrificial oxide. If fabricated from polysilicon, this must be done before the CMOS processing is complete, due to the high temperatures required for depositing and annealing the poly.

However, to minimize chip area it is desirable to fabricate the CMOS drive circuitry underneath the gear region. In this case, the gears and actuators may be fabricated using a low temperature process. One such method has been developed by Zavracky, McGruer and Morrison at Northeastern University, and uses electroplated nickel with a maximum processing temperature of 250 °C. This technique is also advantageous in that the CMOS processing flow does not need to be interrupted, so a standard CMOS process can be used.

The surface of the wafer is directly immersed in the ink reservoir, or in relatively large ink channels. An ultrasonic transducer (for example, a piezoelectric transducer) is positioned in the reservoir. The transducer oscillates the ink pressure at approximately 100 KHz. The ink pressure oscillation is sufficient that ink drops would be ejected from the nozzle were it not blocked by the shutter.

When data signals distributed on the print head indicate that a particular nozzle is to eject a drop of ink, the drive transistor pair of actuators which open the shutter are each pulsed 144 times at around 28 MHz. The peak of the ink pressure variation causes the ink to be squirted out of the nozzle. As the ink pressure goes negative, ink is drawn back into the nozzle, causing drop break-off. The shutter is kept open until the nozzle is refilled on the next positive pressure cycle. It is then shut by a series of 144 pulses to the opposite pair of actuators, to prevent the ink from being withdrawn from the nozzle on the next negative pressure cycle.

Each drop ejection takes two ink pressure cycles, resulting in a 50 KHz drop firing rate.

The nozzle chamber is formed using an anisotropic crystallographic etch of the silicon substrate. The device is manufactured on <100> silicon with a buried boron etch stop layer.

IJ14 VARIABLE RELUCTANCE TAPERED MAGNETIC POLE INK JET

Ink Jet IJ14 is described with reference to the accompanying drawings in which:

Fig. C14.1 illustrates a cross section of a single nozzle and actuator of a 1600 dpi print head. The top includes a two layer pre-stressed nitride spring, the next layer is NiFe, the next is copper. The curved underside to the piston provides a continuously variable reluctance which increases the piston travel.

Fig. C14.2 illustrates an exploded view of the nozzle. Layers from the top are: a) low stress nitride, b) high stress nitride, c) NiFe, d) copper, e) NiFe, f) nitride passivation layer, g) oxide layer containing CMOS metallization, h) silicon layer containing CMOS transistors and the nozzle chamber, i) boron doped silicon etch stop layer containing the nozzle tip.

IJ14 is a direct firing electromagnetic ink jet. A solenoid attracts a soft magnetic piston which ejects the ink. The piston has a tapered inside thickness, providing a variable reluctance as the piston is drawn into the ink chamber.

The use of a tapered magnetic pole allows the travel of the piston to be much greater than a flat moving pole. The taper effectively 'smoothes out' the reluctance change in relation to gap width. When the piston is maximally withdrawn from the chamber, there is still some piston material in close proximity to the fixed pole. Thereby, the force applied to the piston is substantially more linear with distance. The piston is held in position by a pre-stressed spring fabricated from two layers of silicon nitride of different stoichiometry. The bottom layer has more tension than the top layer, causing the spring to curl upwards into a cup shape when released by etching the layers of sacrificial glass used in the fabrication of the nozzle.

When the drive transistor for a nozzle is turned on, the solenoid surrounding the piston is energized, attracting the piston into the chamber. The piston pushes against the ink, causing ink to be squirted out of the nozzle. After approximately 2 μ s, the current to the solenoid is turned off. At the same time, or at a slightly later time, a reverse current is applied, approximately half of the forward current. As the piston will carry some residual magnetism, this causes the piston to move backwards towards its nominal position. The bi-layer spring also helps to return the piston. The reverse current is turned off before the magnetization of the piston is reversed. The return of the piston to its quiescent position causes a low pressure in the ink chamber. This causes ink to begin flowing from the ejected drop back into the nozzle, and also ingests air into the chamber. The forward velocity of the drop and backward velocity of the ink in the chamber are resolved by the ink drop breaking off from the ink in the nozzle. The ink drop then continues to travel towards the recording medium under its own momentum. The nozzle refills due to the surface tension of the ink at the nozzle tip. Shortly after the time of drop breakoff, the meniscus at the nozzle tip is approximately a concave hemisphere. The surface tension exerts a net forward force on the ink, which results in the nozzle refilling.

The repetition rate of the ink jet is principally determined by the nozzle refill time, which will be approximately 100 μ s, depending upon device geometry, ink surface tension, and the volume of the ejected drop.

IJ15 LINEAR SPRING ELECTROMAGNETIC GRILL INK JET

Ink Jet IJ15 is described with reference to the accompanying drawings in which:

Fig. C15.1 illustrates a single nozzle and actuator of a 1600 dpi ink jet viewed from the inside of the ink reservoir. The nozzle actuator is in the quiescent position, with the shutter grill blocking ink access to the nozzle chamber.

Fig. C15.2 illustrates that when energized, the electromagnets attract the bar, causing the linear spring to move the grill sideways, exposing the nozzle chamber.

Fig. C15.3 illustrates an exploded view of the nozzle.

IJ15 is a magnetically actuated ink jet which uses a linear spring to increase the travel of a shutter grill.

An oscillating ink pressure is used to eject ink from the nozzles. Each nozzle has an associated shutter grill, which normally blocks the slots in a fixed grill over the nozzle chamber. The shutter grill is moved so as not to block the fixed grill slots by the electromagnetic actuator whenever an ink drop is to be fired.

The electromagnetic actuator attracts a soft magnetic (NiFe) bar, which is attached to the shutter grill. The bar is also connected in a simple linear spring arrangement to an anchor. The linear spring increases the movement of the shutter by a factor of eight. A one micron motion of the bar towards the electromagnets will result in an eight micron sideways movement. This dramatically improves the efficiency of the system, as the magnetic field falls off strongly with distance, while the spring has a linear relationship between motion in one axis and the other.

The surface of the wafer is directly immersed in the ink reservoir, or in relatively large ink channels. An ultrasonic transducer (for example, a piezoelectric transducer) is positioned in the reservoir. The transducer oscillates the ink pressure at approximately 100 KHz. The ink pressure oscillation is sufficient that ink drops would be ejected from the nozzle were it not blocked by the shutter.

When data signals distributed on the print head indicate that a particular nozzle is to eject a drop of ink, the drive transistor for that nozzle is turned on. This energizes the actuator, which moves the shutter so that it is not blocking the ink chamber. The peak of the ink pressure variation causes the ink to be squirted out of the nozzle. As the ink pressure goes negative, ink is drawn back into the nozzle, causing drop break-off. The shutter is kept open until the nozzle is refilled on the next positive pressure cycle. It is then shut to prevent the ink from being withdrawn from the nozzle on the next negative pressure cycle. Each drop ejection takes two ink pressure cycles, resulting in a 50 KHz drop firing rate.

The nozzle chamber is formed using an anisotropic crystallographic etch of the silicon substrate. Etchant access to the substrate is via the slots in the fixed grill. The device is manufactured on <100> silicon (with a buried boron etch stop layer), but rotated 45° in relation to the <010> and <001> planes. Therefore, the <111> planes which stop the crystallographic etch of the nozzle chamber form a 45° rectangle which superscribes the slots in the fixed grill.

IJ16 LORENZ DIAPHRAGM ELECTROMAGNETIC INK JET

Ink Jet IJ16 is described with reference to the accompanying drawings in which:

Fig. C16.1 illustrates a single nozzle and actuator of a 1600 dpi print head viewed from the inside of the ink reservoir.

Fig. C16.2 illustrates a cross section of the nozzle showing drop ejection.

IJ16 uses the Lorentz force on a current carrying wire in a magnetic field. The static magnetic field is provided by a permanent magnet yoke around the ink jet head.

The actuator consists of a planar copper coil. A section of the coil is embedded in a corrugated diaphragm, which is suspended over a nozzle chamber. The electric current in the diaphragm coil section all flows in one direction. The external magnetic field is in the plane of the chip surface, perpendicular to the current flow in the diaphragm coil. The Lorentz interaction of the diaphragm coil current and the magnetic field results in a force which pushes the diaphragm towards the nozzle, ejecting a drop of ink. The diaphragm is corrugated so that flexure occurs as an elastic bending motion. This is essential, as tensile stress would prevent a flat diaphragm from flexing.

After approximately 3 μ s, the coil current is turned off, and the diaphragm returns to its quiescent position. The diaphragm return 'sucks' some of the ink back into the nozzle, causing the ink ligament connecting the ink drop to the ink in the nozzle to thin. The forward velocity of the drop and backward velocity of the ink in the nozzle cause the ink drop to break off from nozzle ink. The ink drop then continues towards the recording medium. Ink refill of the nozzle chamber is via the two slots at either side of the diaphragm. The ink refill is caused by the surface tension of the ink meniscus at the nozzle, and takes around 100 μ s.

The diaphragm corrugations are formed by depositing resist over a layer of sacrificial glass. The resist is exposed using a halftone mask delineating the corrugation lines. After development, the resist contains thickness corrugations. The resist and sacrificial glass are etched using an etchant which erodes the resist at substantially the same rate as the glass. This transfers the corrugated thickness pattern onto the sacrificial glass. A nitride passivation layer is deposited on the glass. This is followed by a copper

layer, which is patterned using a coil mask. A further nitride passivation layer follows. The slots in the nitride layer at the side of the diaphragm are then etched. This is followed by etching the sacrificial glass.

The nozzle chamber is formed using an anisotropic crystallographic etch of the silicon substrate. Etchant access to the substrate is via the slots at the sides of the diaphragm. The device is manufactured on $\langle 100 \rangle$ silicon (with a buried boron etch stop layer), but rotated 45° in relation to the $\langle 010 \rangle$ and $\langle 001 \rangle$ planes. Therefore, the $\langle 111 \rangle$ planes which stop the crystallographic etch of the nozzle chamber form a 45° rectangle which superscribes the slots in the nitride layer. This etch will proceed quite slowly, due to limited access of etchant to the silicon. However, the etch can be performed at the same time as the bulk silicon etch which thins the wafer.

The drop firing rate is around 7 KHz. The print head has a 'down shooter' configuration.

IJ17 PTFE SURFACE SHOOTING SHUTTERED INK JET

Ink Jet IJ17 is described with reference to the accompanying drawings in which:

Fig. C17.1 illustrates a single nozzle and actuator of a 1600 dpi print head.

Fig. C17.2 illustrates an exploded view of the nozzle, shutter, and actuator.

IJ17 uses an oscillating ink pressure to eject ink from nozzles. Each nozzle has an associated shutter, which normally blocks it. The shutter is moved away from the nozzle opening by thermal bend actuator whenever an ink drop is to be fired.

The nozzles are connected to ink chambers which contain the actuators. These chambers are connected to ink supply channels which are etched through the silicon wafer. The ink supply channels are substantially wider than the nozzles, to reduce the fluidic resistance to the ink pressure wave. The ink channels are connected to an ink reservoir. An ultrasonic transducer (for example, a piezoelectric transducer) is positioned in the reservoir. The transducer oscillates the ink pressure at approximately 100 KHz. The ink pressure oscillation is sufficient that ink drops would be ejected from the nozzle were it not blocked by the shutter.

The shutters are moved by a thermoelastic actuator. The actuators are formed as a coiled serpentine copper heater embedded in polytetrafluoroethylene (PTFE). PTFE has a very high coefficient of thermal expansion (approximately 770×10^{-6}). The current return trace from the heater is also embedded in the PTFE actuator. The current return trace is made wider than the heater trace, and is not serpentine. Therefore, it does not heat the PTFE as much as the serpentine heater does. The serpentine heater is positioned along the inside edge of the PTFE coil, and the return trace is positioned on the outside edge. When actuated, the inside edge becomes hotter than the outside edge, and expands more. This results in the actuator uncoiling.

The heater layer is etched in a serpentine manner both to increase its resistance, and to reduce its effective tensile strength along the length of the actuator. This is so that the low thermal expansion of the copper does not prevent the actuator from expanding in response to the high thermal expansion of the PTFE.

When data signals distributed on the print head indicate that a particular nozzle is to eject a drop of ink, the drive transistor for that nozzle is turned on. This energizes the actuator, which moves the shutter so that it is not blocking the ink chamber. The peak of the ink pressure variation causes the ink to be squirted out of the nozzle. As the ink pressure goes negative, ink is drawn back into the nozzle, causing drop break-off. The shutter is kept open until the nozzle is refilled on the next positive pressure cycle. It is then shut to prevent the ink from being withdrawn from the nozzle on the next negative pressure cycle.

Each drop ejection takes two ink pressure cycles. Half of the nozzles should eject drops in one phase, and the other half of the nozzle should eject drops in the other phase. This minimizes the pressure variations which occur due to different numbers of nozzles being actuated.

The amplitude of the ultrasonic transducer can be altered in response to the viscosity of the ink (which is typically affected by temperature), and the number of drops which are to be ejected in the current cycle. This amplitude adjustment can be used to maintain consistent drop size in varying environmental conditions.

The drop firing rate is around 50 KHz. The ink jet head is suitable for fabrication as a monolithic pagewide print head. The print head has an 'up shooter' configuration.

IJ18 PUSH-PULL BUCKLE STRIP GRILL INK JET

Ink Jet IJ18 is described with reference to the accompanying drawings in which:

Fig. C18.1 illustrates a single nozzle and actuator of a 1600 dpi print head.

Fig. C18.2 illustrates an exploded view of the nozzle, shutter, and actuator.

IJ18 is a fast, extremely low energy ink jet system which uses an oscillating ink pressure to eject ink from nozzles. Each nozzle has an associated fixed grill and moving grill. The slots in the fixed and the bars of the moving grill are normally aligned. This blocks ink flow into the nozzle chamber. When the electrothermal bend actuators are energized, the moving grill shifts so that its slots line up with the slots of the fixed grill. When this occurs ink flow in and out of the nozzle chamber is relatively unobscured.

The grill is moved by a pair of thermoelastic actuators. One of the actuators pushes the grill, and the other pulls it. The actuators contain serpentine metal wires which are mechanically and electrically connected to the substrate at both ends. The four point mechanical connection gives the grill a high degree of stability and rigidity. The serpentine copper heaters of the actuators are embedded in polytetrafluoroethylene (PTFE). PTFE has a very high coefficient of thermal expansion (approximately 770×10^{-6}). The serpentine heater is positioned along one edge of the PTFE actuator. When actuated, this edge becomes hotter than the opposite edge, and expands more. This results in the actuator bending. The two actuators are arranged so that both bend in the same direction. The movable grill is fabricated from the same PTFE layer as the actuators, and is integrally connected between the centers of the two actuators. The heater layer is etched in a serpentine manner both to increase its resistance, and to reduce its effective tensile strength along the length of the actuator. This is so that the low thermal expansion of the copper does not prevent the actuator from expanding in response to the high thermal expansion of the PTFE. The PTFE is made hydrophilic so that the nozzle fills by capillarity.

The grills lie between ink channels which are etched through the silicon wafers, and nozzle chambers which are fabricated as hollow structures of silicon nitride. The ink supply channels are substantially wider than the nozzles, to reduce the fluidic resistance to the ink pressure wave. The ink channels are connected to an ink reservoir. An ultrasonic transducer (for example, a piezoelectric transducer) is positioned in the reservoir. The transducer oscillates the ink pressure at approximately 100 KHz. The ink pressure oscillation is sufficient that ink drops would be ejected from the nozzle were it not blocked by the shutter.

When a drop is to be printed, the actuators are energized, moving the shutter so that it does not block the ink chamber. The peak of the ink pressure variation causes the ink to be squirted out of the nozzle. As the ink pressure goes negative, ink is drawn back into the nozzle, causing drop break-off. The shutter is kept open until the nozzle is refilled on the next positive pressure cycle. It is then shut to prevent the ink from being withdrawn from the nozzle on the next negative pressure cycle.

The drop firing rate is around 50 KHz. The ink jet head is suitable for fabrication as a monolithic pagewide print head. The print head has an 'up shooter' configuration.

IJ19 THERMOELASTIC BEND ACTUATOR SHUTTERED INK JET

Ink Jet IJ19 is described with reference to the accompanying drawings in which:

Fig. C19.1 illustrates a pair of nozzles and actuators of a 1200 dpi print head viewed from the inside of the ink reservoir. The closer nozzle is open, and the further nozzle is closed.

Fig. C19.2 illustrates an exploded view of the pair of nozzles.

IJ19 uses an oscillating ink pressure to eject ink from nozzles. Each nozzle has an associated shutter and fixed grill, made from polysilicon. There are two polysilicon thermoelastic bend actuators for each nozzle. One of these actuators is mechanically connected to the shutter, and moves it into the 'open' position whenever a drop is to be fired. The other actuator is much smaller, and forms a 'catch' to hold the shutter open during the drop ejection cycle. Energizing the catch actuator causes it to bend away from a notch formed in the shutter, releasing the shutter and allowing it to return to the closed position.

The hot arm of each bend actuator is coated with a polymer which has a poor thermal conductivity. The cold arm of each bend actuator is in thermal contact with the water based ink.

The nozzles are connected to ink chambers, which in turn are connected to an ink reservoir. An ultrasonic transducer (typically a piezo electric transducer) is positioned in the reservoir. The transducer oscillates the ink pressure at approximately 100 KHz. The ink pressure oscillation is such that ink drops would be ejected from the nozzle.

Ink drop ejection takes two cycles of the pressure wave. The shutter actuator is energized at the beginning of the positive pressure phase of the first cycle. The shutter moves into the open position, and the catch locks it in position. After a short period, the power to the shutter actuator is turned off, and the hot arm of the actuator begins to cool. Meanwhile, the high ink pressure forces a drop of ink from the nozzle. The ink pressure then goes into the negative pressure phase, whereupon ink around the base of the ejected drop is sucked back into the nozzle, along with an air bubble. This causes the ejected ink drop to break off from the ink in the nozzle. The shutter is kept open until the nozzle is refilled on the next positive pressure cycle. At the end of the refill half-cycle, the catch actuator is energized. This allows the shutter to rapidly return to its nominal position, preventing the ink from being withdrawn from the nozzle on the next negative pressure cycle.

The catch has two purposes: to reduce power consumption by holding the shutter open without requiring power to the main actuator, and to allow the main actuator time to cool down before it is to shut, thereby achieving a rapid closing of the shutter.

The amplitude of the ultrasonic transducer can be altered in response to the viscosity of the ink (which is typically affected by temperature), and the number of drops which are to be ejected in the current cycle.

The drop firing rate is around 50 KHz. The ink jet head is suitable for fabrication as a monolithic pagewide print head. The print head has a 'down shooter' configuration.

IJ20 CURLING CALYX THERMOELASTIC INK JET

Ink Jet IJ20 is described with reference to the accompanying drawings in which:

Fig. C20.1 illustrates a cross section of a single nozzle showing 4 of the 8 actuator 'petals'.

Fig. C20.2 illustrates when the actuator is energized, the 'petals' curl into a calyx formation. Ink above the actuator is trapped by the nozzle chamber wall, and is expelled from the nozzle. Ink flows in under the nozzle chamber wall to fill the expanding region under the actuator.

Fig. C20.3 illustrates an exploded view of the nozzle.

IJ20 has one thermoelastic bend actuator for each nozzle. The center of the actuator is mechanically and electrically connected to the substrate. Around the center are eight petal shaped thermoelastic bend actuators. These 'petals' are wired in series, and are all energized simultaneously. Each 'petal' contains an electrothermal heater composed of copper or other electrically conducting material. The copper heater is embedded in a polytetrafluoroethylene (PTFE) paddle. PTFE is used because it has a very high coefficient of thermal expansion.

When data signals distributed on the print head indicate that a particular nozzle is to eject a drop of ink, the drive transistor for that nozzle is turned on for around 3 μ s. This energizes the heaters in the 'petals' which heat the lower half of each petal. The top half is not heated, and is cooled by the water based ink. The lower PTFE expands rapidly. This expansion causes the petals to curl away from the substrate into a calyx formation, pushing ink out of the nozzle.

When the heater current is turned off, the petals return to their quiescent position. This 'sucks' some of the ink back into the nozzle, causing the ink drop to break off from the ink in the nozzle. The ink drop then continues towards the recording medium.

The nozzle is fabricated as a hole in an integrated nozzle plate of silicon nitride. The nozzle plate is raised from the substrate to form the nozzles and their chambers by the use of a sacrificial oxide. Before deposition of the nitride, the sacrificial oxide is etched to two different depths. A circular rim is etched to the depth of the actuator to form the suspended nozzle chamber. A series of holes are also

etched right through the sacrificial oxide. When filled with nitride these form posts which suspend the nozzle plate above the actuators.

The top surface of the PTFE is treated to make it hydrophilic. The lower surface can be left hydrophobic. In this case, an air bubble will form under the petals, as the nozzle will not completely fill by capillarity.

The air bubble prevents the water based ink contacting the underside of the paddle, achieving a higher temperature with lower power.

The drop firing rate is around 7 KHz, limited by the nozzle refill time. The ink jet head is suitable for fabrication as a monolithic pagewide print head.

IJ21 SHUTTERED OSCILLATING PRESSURE INK JET

Ink Jet IJ21 is described with reference to the accompanying drawings in which:

Fig. C21.1 illustrates a single nozzle and actuator of a 1600 dpi print head viewed from the inside of the ink reservoir.

Fig. C21.2 illustrates an exploded view of the nozzle.

IJ21 uses an oscillating ink pressure to eject ink from nozzles. Each nozzle has an associated shutter, which is moved by an electrothermal bend actuator. The shutter is normally open, and is moved to cover the ink chamber whenever an ink drop is to be prevented from being ejected.

The nozzles are connected to ink chambers, which in turn are connected to an ink reservoir. An ultrasonic transducer (typically a piezo electric transducer) is positioned in the reservoir. The transducer oscillates the ink pressure at approximately 100 KHz. The ink pressure oscillation is sufficient to eject ink drops from the nozzle on every cycle if the shutter is open.

The coiled actuator is constructed from laminated conductors of either differing resistivities, different cross sectional areas, different indices of thermal expansion, different thermal conductivities to the ink, different lengths, or some combination thereof. The coiling radius of the actuator changes when a current is passed through it, as one side of the coiled beam expands differently than the other.

The actuator controls the position of the shutter, so that it can cover none, all or part of the nozzle chamber. If the shutter covers none of the nozzle chamber, then the oscillating ink pressure is transmitted to the nozzle. If the shutter covers the ink chamber, then the oscillating ink pressure is significantly attenuated at the nozzle. The ink pressure variations are not entirely stopped, due to leakage around the shutter, and flexing of the shutter under varying pressure. The shutter may also be part way across the ink chamber, resulting in partial attenuation of the ink pressure variations. This may be used to vary the volume of the ejected drop. Drop volume control may be used either to implement a degree of continuous tone operation, to regulate the drop volume, or both.

When data signals distributed on the print head indicate that a particular nozzle is to eject a drop of ink, the drive transistor for that nozzle is turned off. This de-energizes the actuator, allowing the shutter to move so that it is not blocking the ink chamber. The peak of the ink pressure variation causes the ink to be squirted out of the nozzle. As the ink pressure goes negative, ink is drawn back into the nozzle, causing drop break-off. The shutter is left open until the nozzle is refilled on the next positive pressure cycle. It is then shut to prevent the ink from being withdrawn from the nozzle on the next negative pressure cycle. Each drop ejection takes two ink pressure cycles.

The drop firing rate is around 50 KHz. The ink jet head is suitable for fabrication as a monolithic pagewide print head. The print head has a 'down shooter' configuration.

IJ22 IRIS MOVEMENT THERMOELASTIC BEND ACTUATOR INK JET

Ink Jet IJ22 is described with reference to the accompanying drawings in which:

Fig. C22.1 illustrates a single nozzle and actuator of a 1600 dpi surface shooting print head.

Fig. C22.2 illustrates a view of the nozzle in quiescent position, with the nozzle plate removed.

Fig. C22.3 illustrates a view of the nozzle in energized position, with the nozzle plate removed.

IJ22 has four thermoelastic bend actuators for each nozzle. The actuators are energized simultaneously. Attached to each actuator is a curved upright vane. When energized, the actuators cause the vanes to contract in a manner similar to a camera iris. This contraction reduces the volume contained by the vanes, causing ink to be ejected from the nozzle. When power to the actuators is turned off, they cool down rapidly, returning to their normal position. As they return, the pressure in the space enclosed by the vanes falls, causing some of the ink to be 'sucked' back into the nozzle. This results in the ejected drop breaking off from the ink in the nozzle, and travelling towards the print medium.

The actuators each have two arms, an expanding, flexible arm, and a rigid arm. Both arms are fixed at one end, and are joined together at the other end. The thermally expanding arms are formed of a serpentine copper heater embedded in polytetrafluoroethylene (PTFE). The heater layer is etched in a serpentine manner both to increase its resistance, and to reduce its effective tensile strength along the length of the actuator. This is so that the low thermal expansion of the copper does not prevent the actuator from expanding in response to the high thermal expansion of the PTFE. The rigid arm comprises the return trace of the copper heater, and the vane. The vane is not present along the entire length of the rigid arm. Approximately 20% of the length of the rigid arm has no vane, and is flexible. This section of the rigid arm bends in response to the force created by the elongation of the expanding arm. This localization of the bending results in a greater overall movement of the vane.

Ink flow to the nozzle is across the chip surface. For low speed printers this can be supplied via the chip edge. For higher speed operation, ink channels should be etched through the substrate to increase the ink supply.

The drop firing rate is around 7 KHz. The printing speed is primarily constrained by the nozzle refill time, which is approximately 100 μ s if the ink is not under pressure. The ink jet head is suitable for fabrication as a monolithic pagewide print head.

IJ23 DOWN SHOOTER WITH PTFE THERMOELASTIC BEND ACTUATOR

Ink Jet IJ23 is described with reference to the accompanying drawings in which:

Fig. C23.1 illustrates a cross section of a single nozzle and actuator of a 1600 dpi print head, shown during the firing of a drop.

Fig. C23.2 illustrated an exploded view of the nozzle.

IJ23 has one thermoelastic bend actuator for each nozzle. One end of the actuator is mechanically connected to the substrate, and the other end is connected to a stiff paddle. When energized, the actuator bends into a nozzle chamber in the substrate. This motion causes the attached paddle to eject a drop of ink through a nozzle at the opposite side of the nozzle chamber. As the actuator is cooled by the ink, it returns to its nominal position.

The actuator is composed of a copper heater embedded in a polytetrafluoroethylene (PTFE) paddle. PTFE is used because it has a very high coefficient of thermal expansion.

The bottom layer of PTFE is thick (around 3 μ m) and is not significantly heated by the heater in the brief time that the heater is energized. The copper heater is formed as a serpentine wire of approximately 0.3 μ m thickness. The copper is deposited over thickness corrugations in the bottom PTFE layer, and then etched in the serpentine pattern. The PTFE corrugations can be formed by exposing a resist layer to a halftone mask, then etching the resist and the underlying PTFE at the same rate. This is done to increase the response speed of the actuator, by reducing the thermal distance between the heater and the PTFE that is to be heated. After heater deposition and etching, a further 0.5 μ m of PTFE is deposited to make the hot side of the bend actuator. The PTFE is treated to make it hydrophilic.

When data signals distributed on the print head indicate that a particular nozzle is to eject a drop of ink, the drive transistor for that nozzle is turned on. This energizes the heater in the paddle for that nozzle. The heater is energized for approximately 3 μ s, with the actual duration depending upon the design chosen for the actuator and nozzle. The heater heats the upper PTFE layer, but does not have time to significantly heat the lower thick PTFE layer. This expansion causes the paddle to bend, pushing ink out of the nozzle.

When the heater current is turned off, the paddle begins to return to its quiescent position. The paddle return 'sucks' some of the ink back into the nozzle, causing the ink drop to break off from the ink in the nozzle. The ink drop then continues towards the recording medium.

The drop firing rate is around 7 KHz. The ink jet head is suitable for fabrication as a monolithic pagewide print head. The print head has a 'down shooter' configuration.

IJ24 CONDUCTIVE PTFE THERMOELASTIC BEND ACTUATOR INK JET

Ink Jet IJ24 is described with reference to the accompanying drawings in which:

Fig. C24.1 illustrates a single nozzle and actuator of a 1600 dpi print head.

Fig. C24.2 illustrates an exploded view of the nozzle.

Fig. C24.3 illustrates two nozzle chambers, showing the air vent between them.

IJ24 has one thermoelastic bend actuator for each nozzle. The bend actuator is constructed from a dual layer of PTFE. The bottom layer is doped to become conductive, forming an electrothermal heater. One end of the actuator is mechanically connected to the substrate, and the other end is connected to a stiff paddle. When energized, the actuator bends away from the substrate towards a nozzle, causing the attached paddle to eject ink. As the actuator is cooled by the ink, it returns to its nominal position. The nozzle is fabricated as a hole in an integrated nozzle plate of silicon nitride. The nozzle plate is raised from the substrate to form the ink channels, nozzles, filters, and nozzle chambers by the use of a sacrificial oxide.

The actuator is composed of two layers:

- 1) A conductive polytetrafluoroethylene (PTFE) lower layer, of around 1 μm thickness. PTFE has a very high coefficient of thermal expansion (approximately 770×10^{-6} , or around 380 times that of silicon). The PTFE is made conductive by dispersing conductive material in the polymer matrix. For example, a dispersion of about 2% of single wall carbon nanotubes of average length of about 1 μm . The resistivity required for convenient low voltage operation is around 100 $\mu\Omega\text{m}$.
- 2) A PTFE upper layer of around 3.5 μm thickness. The upper surface of the PTFE is treated to make it hydrophilic. This may be achieved by exposing the PTFE surface to a high energy plasma in high vacuum, with ammonia inlet. The plasma breaks the surface PTFE chains, the ends of which are then available to react with the ammonia, forming partial positive charges on terminal NH_2 groups, and negative charges on the fluorine.

When a drop is to be fired, the drive transistor for that nozzle is turned on for around 4 μs . This energizes the lower (conductive) PTFE, wherein electrical resistive losses are converted to heat. The top PTFE is not heated, and is cooled by the water based ink. The lower PTFE expands rapidly. This expansion causes the actuator to bend, moving the paddle which pushes ink out of the nozzle. There is an air bubble between the paddle and the substrate, which forms due to the hydrophobic nature of the PTFE on the back surface of the paddle. An air vent connects the air bubble to the ambient air, allowing the paddle to move without being held back by a large pressure drop under the paddle. The air bubble also prevents the water based ink contacting the underside of the paddle, achieving a higher temperature with lower power.

When the heater current is turned off, the paddle begins to return to its quiescent position. The paddle return causes the ink drop to break off from the ink in the nozzle. The ink drop then continues towards the recording medium.

IJ25 MAGNETOSTRICTIVE INK JET

Ink Jet IJ25 is described with reference to the accompanying drawings in which:

Fig. C25.1 illustrates a view of a single 1600 dpi actuator. The magnetostrictive material (Terfenol-D) is surrounded by a copper coil. Below the magnetostrictive paddle is the nozzle chamber, which includes the nozzle.

Fig. C25.2 illustrates a cross section of the nozzle showing the copper coil, the Terfenol-D layer which forms the bend actuator of the paddle, a nitride layer which provides a

bend 'foil' of high tensile strength, a crystallographically etched nozzle chamber in silicon, and a nozzle membrane fabricated in boron doped silicon.

IJ25 uses bend actuator which operates on magnetostrictive principles. The actuator uses the giant magnetostrictive effect of materials such as Terfenol-D (an alloy of terbium, dysprosium and iron developed at the Naval Ordnance Laboratory, hence Ter-Fe-NOL).

The actuator is suspended at the entrance to a nozzle chamber, on the opposite side to the nozzle. The actuator is in the center of a coil of copper, which forms a solenoid. Copper is chosen for the solenoid both for its low resistivity and its high resistance to electromigration. The copper coil is directly cooled by the ink, so relatively large electrical currents can be accommodated without a significant temperature rise. When the solenoid is energized, the resultant magnetic field causes the Terfenol-D layer of the actuator to expand, the actuator bends into the nozzle chamber, and pushes ink out of the nozzle.

There is one magnetostrictive bend actuator for each nozzle. One end of the bend actuator is mechanically connected to the substrate. The other end is free to move.

The actuator is composed of three layers:

- 1) An silicon nitride bottom layer. This layer is of high stiffness, which is deposited stress free with the appropriate stoichiometry. Its purpose is to prevent the actuator from elongating, so that the expansion of the magnetostrictive layer is manifested as a bending motion.
- 2) A magnetostrictive layer. A material with giant magnetostrictive properties such as Terfenol-D is used. The layer thickness is chosen to have approximately the same tensile strength as the thick nitride layer.
- 3) A thin silicon nitride passivation layer, to prevent corrosion of the Terfenol-D. This is actually made of two layers: the nitride insulation between the Terfenol-D and the coil, and the coil passivation layer. The total thickness should be less than a third of the thickness of the bottom nitride layer.

The drop firing rate is around 7 KHz, constrained by the nozzle refill time. The ink jet head is suitable for fabrication as a monolithic pagewidth print head, which has a 'down shooter' configuration.

IJ26 SHAPE MEMORY ALLOY INK JET

Ink Jet IJ26 is described with reference to the accompanying drawings in which:

Fig. C26.1 illustrates a single nozzle and actuator of a 1600 dpi print head in the quiescent position. The nozzle is viewed from the inside of the ink reservoir.

Fig. C26.2 illustrates a cross section of the nozzle firing a drop. The SMA actuator is in its austenitic phase due to being electrothermally heated.

Fig. C26.3 illustrates how the SMA actuator returns to its martensitic state as it cools down. Elastic stress in the nitride layer returns it to a bent shape.

IJ26 relies upon the thermal transition of a shape memory alloy (SMA) from its martensitic phase to its austenitic phase. The thermal transition is achieved by passing an electrical current through the SMA. The actuator is suspended at the entrance to a nozzle chamber, on the opposite side to the nozzle. The actuator is bent away from the nozzle when in its quiescent state. When energized, the actuator straightens, and pushes ink out of the nozzle. Energizing the actuator requires supplying enough energy to raise the SMA above the transition temperature, and to provide the latent heat of transformation to the SMA.

The SMA martensitic phase must be pre-stressed to achieve a different shape from the austenitic phase. For print heads with many thousands of nozzles, it is important to achieve this re-stressing in a bulk manner. IJ26 achieves this by depositing a layer of silicon nitride using PECVD at around 300 °C over the SMA. This deposition occurs while the SMA is in the austenitic shape. After the print head cools to room temperature, the substrate under the SMA bend actuator is removed by chemical etching. The silicon nitride is under tensile stress, and causes the actuator to curl upwards. The weak martensitic phase of the SMA provides little resistance to this curl. When the SMA is heated to its austenitic phase, it returns to the flat shape into which it was annealed during the nitride deposition.

There is one SMA bend actuator for each nozzle. One end of the SMA bend actuator is mechanically connected to the substrate. The other end is free to move under the stresses inherent in the layers.

The nozzle plate is formed from a buried etch stop layer in the silicon substrate.

The actuator is composed of three layers:

- 1) An SiO₂ lower layer. This layer acts as a stress 'reference' for the nitride tensile layer. It also protects the SMA from the crystallographic silicon etch that forms the nozzle chamber. This layer is formed as part of the standard CMOS process for the active electronics of the print head.
- 2) A SMA heater layer. A SMA such as nickel titanium (NiTi) alloy is deposited and etched into a serpentine form to increase the electrical resistance.
- 3) An silicon nitride top layer. This is a thin layer is of high stiffness which is deposited using PECVD. The nitride stoichiometry is adjusted to achieve a layer with significant tensile stress at room temperature relative to the SiO₂ lower layer. Its purpose is bend the actuator at the low temperature martensitic phase.

The drop firing rate is around 7 KHz. The ink jet head is suitable for fabrication as a monolithic pagewide print head. The print head has a 'down shooter' configuration.

IJ27 THERMOELASTIC BUCKLE PLATE SURFACE SHOOTER INK JET

Ink Jet IJ27 is described with reference to the accompanying drawings in which:

Fig. C27.1 illustrates a single nozzle and actuator of a 1600 dpi print head.

Fig. C27.2 illustrates an exploded view of the nozzle. Two layers of PTFE are shown. The top layer is the buckle plate actuator, and the bottom layer is a hydrophobic vent grill which allows air into the space under the buckle plate.

IJ27 has one thermoelastic buckle plate actuator for each nozzle. Both ends of the actuator is mechanically connected to the substrate. When energized, a heater embedded in the actuator causes the actuator to expand. As both ends are fixed, the expansion causes a buckling in the center. To ensure that the buckling always occurs upwards, the heater is positioned closer to the top surface of the actuator than the bottom surface. As the buckle plate bends upwards, it pushes ink out of the nozzle. The power to the actuator is maintained for approximately 3 μ s.

There is an air bubble between the buckle plate and the substrate, which forms due to the hydrophobic nature of the PTFE on the back surface of the buckle plate. An air vent connects the air bubble to the ambient air, allowing the buckle plate to move without being held back by a reduction in air pressure as the bubble expands. The air bubble also prevents the water based ink contacting the underside of the buckle plate, achieving a higher temperature with lower power.

When the heater current is turned off, the actuator is rapidly cooled by the ink. The buckle plate begins to return to its quiescent position. The buckle plate return 'sucks' some of the ink back into the nozzle, causing the ink drop to break off from the ink in the nozzle. The ink drop then continues towards the recording medium. The return of the buckle plate forces the air under the buckle plate back out of the vents.

The nozzle is fabricated as a hole in an integrated nozzle plate of silicon nitride. The nozzle plate is raised from the substrate to form the ink channels, nozzles, filters, and nozzle chambers by the use of a sacrificial oxide.

The actuator is composed of a copper heater embedded in a polytetrafluoroethylene (PTFE) buckle plate. PTFE is used because it has a very high coefficient of thermal expansion. The copper heater is formed as a serpentine wire of approximately 0.3 μ m thickness. The upper surface of the PTFE is treated to make it hydrophilic.

The drop firing rate is around 7 KHz. The ink jet head is suitable for fabrication as a monolithic pagewide print head.

IJ28 THERMOELASTIC ROTARY IMPELLER SURFACE SHOOTING INK JET

Ink Jet IJ28 is described with reference to the accompanying drawings in which:

Fig. C28.1 illustrates a single nozzle and actuator of a 1600 dpi surface shooting print head.

Fig. C28.2 illustrates an array of nozzles for a single ink color. The spacing between nozzles is $64\ \mu\text{m}$, so four rows are required for 1600 dpi operation (with $16\ \mu\text{m}$ dot spacing). The holes between the nozzle rows are ink channels, etched completely through the wafer. Ink is supplied to the print head at the back of the wafer.

IJ28 has one rotary impeller for each nozzle. The rotary impeller has a set of moving vanes and a set of fixed vanes. When actuated, the set of moving vanes rotates towards the fixed vanes, displacing the ink between the vanes. The vanes are arranged so that the path of least resistance for the displaced ink is out of the nozzle.

The fixed vanes of the impeller are connected to the nozzle plate, which is suspended above the substrate on fixed posts. The moving vanes are attached to a copper rotating frame. One end of each of two thermoelastic expansion actuators are attached to the rotating frame. The actuators are connected to the rotating frame tangentially along the outer rim of the frame, in opposite directions.

The actuators are formed of a serpentine copper heater embedded in polytetrafluoroethylene (PTFE). The heater layer is etched in a serpentine manner both to increase its resistance, and to reduce its effective tensile strength along the length of the actuator. This is so that the low thermal expansion of the copper does not prevent the actuator from expanding in response to the high thermal expansion of the PTFE.

The copper heater forms a continuous circuit through both actuators, via the rotating frame. The other ends of the actuators are mechanically connected to the substrate, and electrically connected to the drive circuitry.

When data signals distributed on the print head indicate that a particular nozzle is to eject a drop of ink, the drive transistor for that nozzle is turned on. Both actuators are energized simultaneously.

The heater heats the PTFE layer, which expands. This expansion causes the actuators to elongate. The two actuators expand in opposite directions, and are connected to opposite sides of the rotating frame. The resultant forces on the frame cause a rotation of approximately 30 degrees.

When the heater current is turned off, the actuators return to their quiescent positions, rotating the frame back to its normal position.

The return of the frame rotates the moving vanes away from the fixed vanes. This 'sucks' some of the ink back into the nozzle, causing the ink ligament connecting the ink drop to the ink in the nozzle to thin. The forward velocity of the drop and backward velocity of the ink in the nozzle are resolved by the ink drop breaking off from the ink in the nozzle. The ink drop then continues towards the recording medium.

The actuator and impeller are at the quiescent position until the next drop ejection cycle.

The drop firing rate is around 7 KHz. The ink jet head is suitable for fabrication as a monolithic pagewide print head. The print head has a 'roof shooter' configuration.

IJ29 PTFE-BASED THERMOELASTIC BEND ACTUATOR INK JET

Ink Jet IJ29 is described with reference to the accompanying drawings in which:

Fig. C29.1 illustrates a single nozzle and actuator of a 1600 dpi print head.

Fig. C29.2 illustrates an array of nozzles for a single ink color. The spacing between nozzles is $32\ \mu\text{m}$, so two rows are required for 1600 dpi operation (with $16\ \mu\text{m}$ dot spacing). The holes between the nozzle rows are ink channels, etched completely through the wafer.

IJ29 has one thermoelastic bend actuator for each nozzle. One end of the actuator is mechanically connected to the substrate, and the other end is connected to a stiff paddle. When energized, the actuator bends away from the substrate towards a nozzle, causing the attached paddle to eject ink. As the actuator is cooled by the ink, it returns to its nominal position. The nozzle is fabricated as a hole in an integrated nozzle plate of silicon nitride. The nozzle plate is raised from the substrate to form the ink channels.

nozzles, filters, and nozzle chambers by the use of a sacrificial oxide. This is to avoid the requirement for separate fabrication and high precision assembly of a substrate and a nozzle plate.

The actuator contains four layers:

- 1) A polytetrafluoroethylene (PTFE) lower layer. PTFE has a very high coefficient of thermal expansion (approximately 770×10^{-6} , or around 380 times that of silicon).
- 2) A heater layer. A serpentine heater is etched in this layer, which may be Nichrome, copper or other suitable material with a resistivity such that the drive voltage for the heater is compatible with the drive transistors. The heater is formed in serpentine fashion so as to have very little tensile strength in the direction along the length of the actuator.
- 3) A PTFE upper layer. This layer also expands when heated by the heater.
- 4) A silicon nitride layer. This is a thin layer is of high stiffness and low coefficient of thermal expansion. Its purpose is to ensure that the actuator bends, instead of simply elongating. Silicon nitride is used because it is a standard semiconductor material, and SiO_2 cannot easily be used if it is also the sacrificial material.

When a drop is to be fired, the drive transistor is turned on, energizing the heater for around 3 μs . The heater heats the PTFE layer, which expands rapidly. This expansion causes the actuator to bend, moving the paddle, and thereby pushing ink out of the nozzle. There is an air bubble between the paddle and the substrate, which forms due to the hydrophobic nature of the PTFE on the back surface of the paddle. This air bubble reduces the thermal coupling to the hot side of the actuator, achieving a higher temperature with lower power. The cold side of the actuator is cooled by the water based ink. The presence of the air bubble also means that less ink is required to move under the paddle when the actuator is energized. These factors lead to a lower power consumption of the actuator.

When the heater current is turned off, the paddle begins to return to its quiescent position. The paddle return 'sucks' some of the ink back into the nozzle, causing the ink drop to break off from the ink in the nozzle.

The drop firing rate is around 7 KHz.

IJ30 PTFE AND CORRUGATED COPPER THERMOELASTIC BEND ACTUATOR

Ink Jet IJ30 is described with reference to the accompanying drawings in which:

Fig. C30.1 illustrates a cross section of a single nozzle and actuator of a 1600 dpi print head, shown during the firing of a drop. When the paddle is actuated, air vents in through holes in a hydrophobic PTFE grill under the paddle. This prevents a low pressure region forming under the paddle and holding it back.

Fig. C30.2 illustrates an exploded view of the nozzle.

Fig. C30.3 illustrates detail of the doubly serpentine heater.

IJ30 has one thermoelastic bend actuator for each nozzle. One end of the actuator is mechanically connected to the substrate, and the other end is connected to a stiff paddle. When energized, the actuator bends away from the substrate towards a nozzle, causing the attached paddle to eject ink. As the actuator is cooled by the ink, it returns to its nominal position. The nozzle is fabricated as a hole in an integrated nozzle plate of silicon nitride. The nozzle plate is raised from the substrate to form the ink channels, nozzles, filters, and nozzle chambers by the use of a sacrificial oxide.

The actuator is composed of a copper heater embedded in a polytetrafluoroethylene (PTFE) paddle. PTFE is used because it has a very high coefficient of thermal expansion.

The copper heater is formed as a serpentine wire of approximately 0.3 μm thickness. The copper is deposited over thickness corrugations of the PTFE, and then etched in the serpentine pattern. The PTFE corrugations can be formed by exposing a resist layer to a halftone mask, then etching the resist and the underlying PTFE at the same rate. As a result the copper heater 'winds' throughout the volume of the PTFE. This is done to increase the response speed of the actuator, by reducing the thermal distance between the heater and the PTFE of the actuator. After heater deposition and etching, a further 3 μm of PTFE is deposited to make the cold side of the bend actuator. The upper surface of the PTFE is treated to make it hydrophilic.

When data signals distributed on the print head indicate that a particular nozzle is to eject a drop of ink, the drive transistor for that nozzle is turned on for around 3 μs . This energizes the heater which

heats the lower half of the PTFE. The top half is not heated, and is cooled by the water based ink. The lower PTFE expands rapidly. This expansion causes the actuator to bend, moving the paddle which pushes ink out of the nozzle. There is an air bubble between the paddle and the substrate, which forms due to the hydrophobic nature of the PTFE on the back surface of the paddle. An air vent connects the air bubble to the ambient air, allowing the paddle to move without being held back by a large pressure difference under the paddle. The air bubble also prevents the water based ink contacting the underside of the paddle, achieving a higher temperature with lower power.

When the heater current is turned off, the paddle begins to return to its quiescent position. The paddle return 'sucks' some of the ink back into the nozzle, causing the ink drop to break off from the ink in the nozzle. The ink drop then continues towards the recording medium. The return of the paddle forces the air under the paddle back out of the vents.

The drop firing rate is around 7 KHz. The ink jet head is suitable for fabrication as a monolithic pagewide print head.

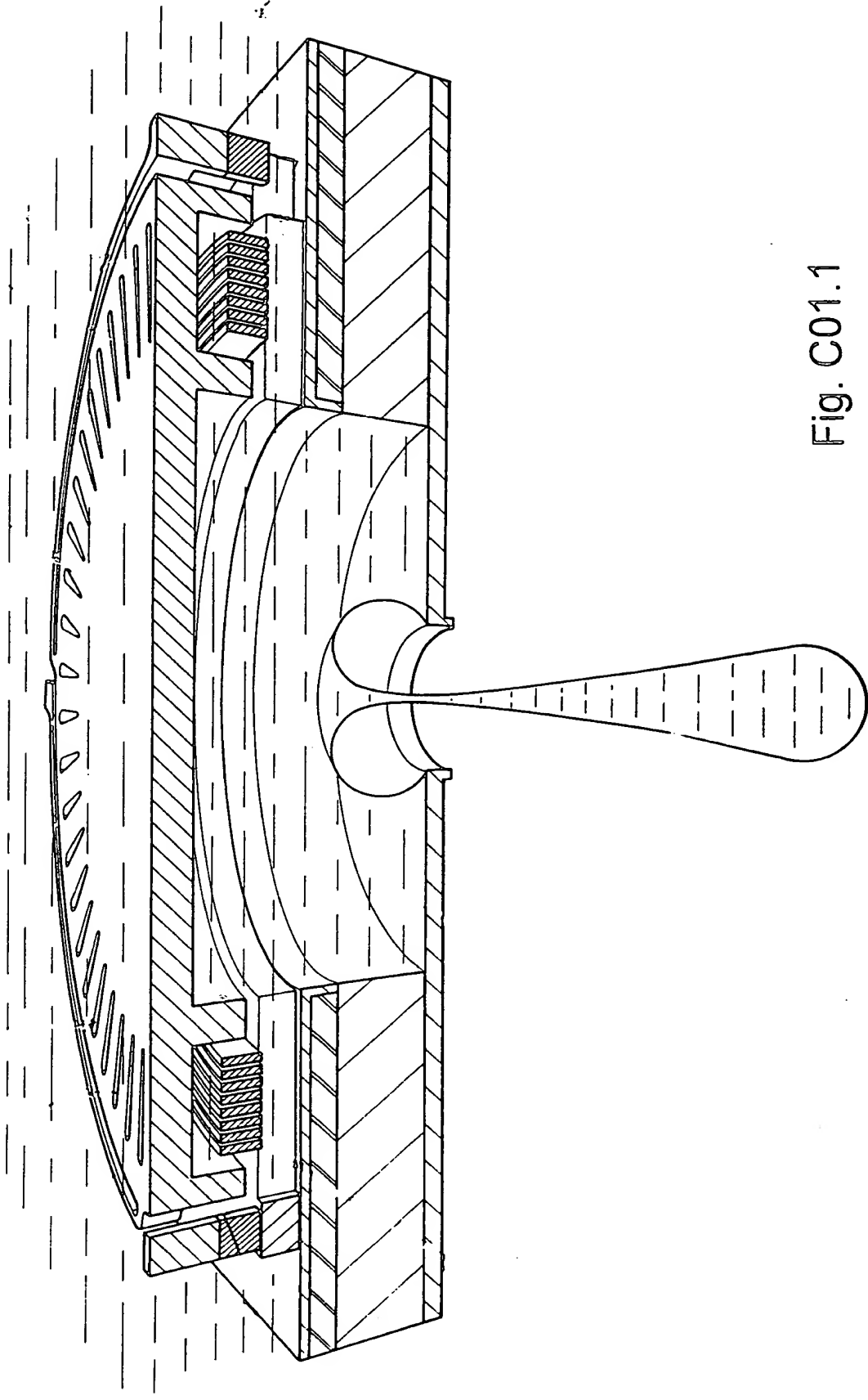


Fig. C01.1

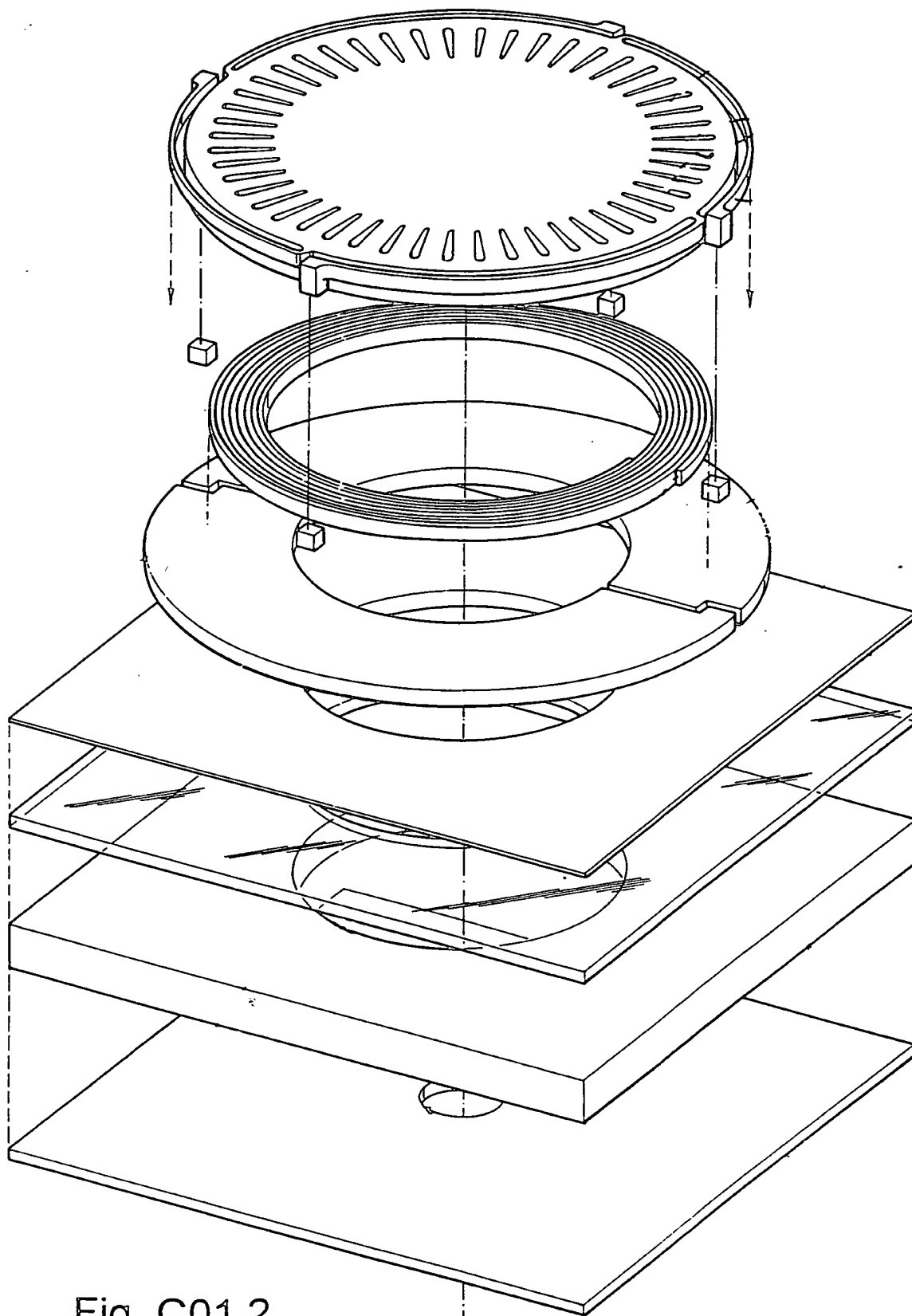


Fig. C01.2

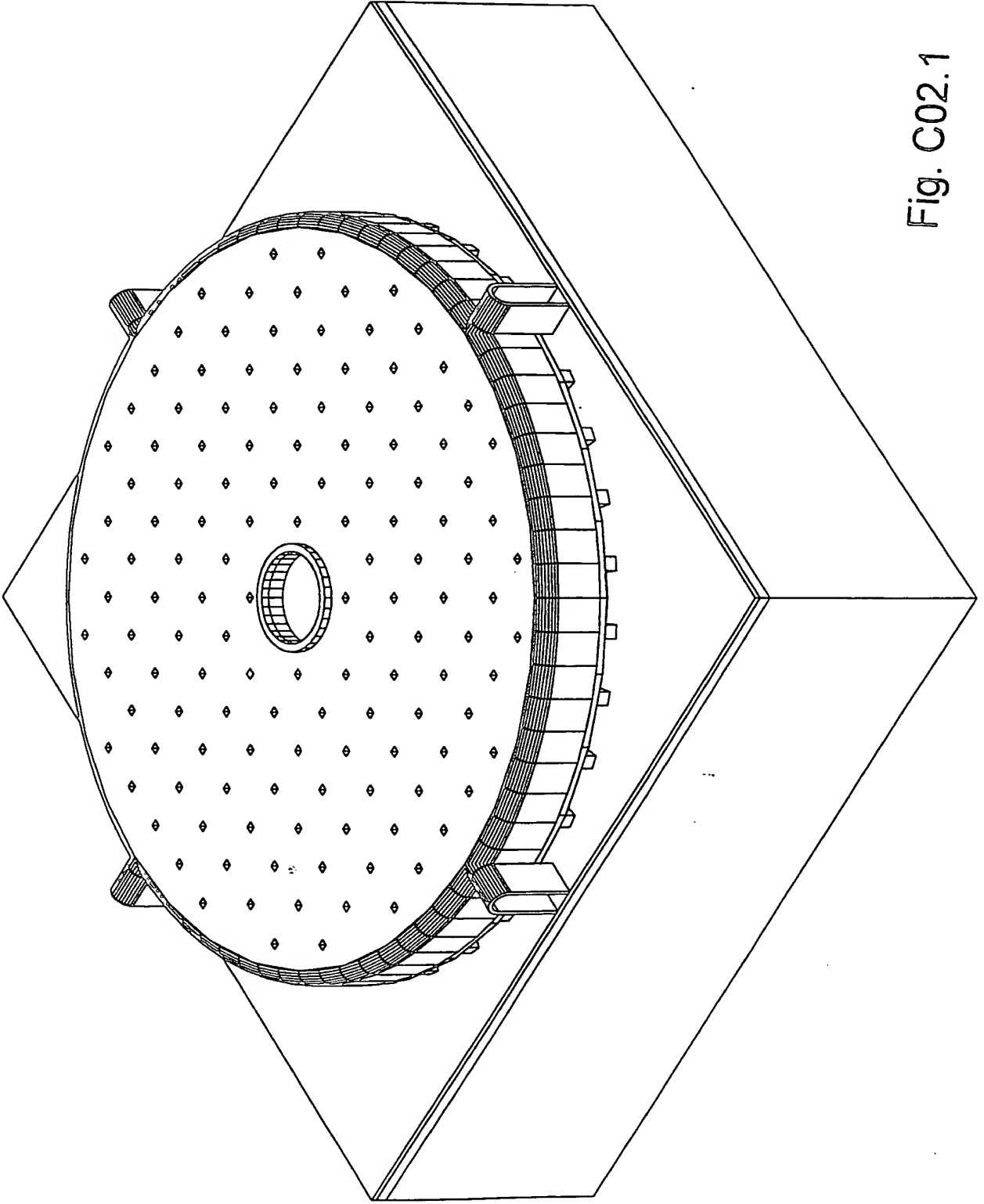


Fig. C02.1

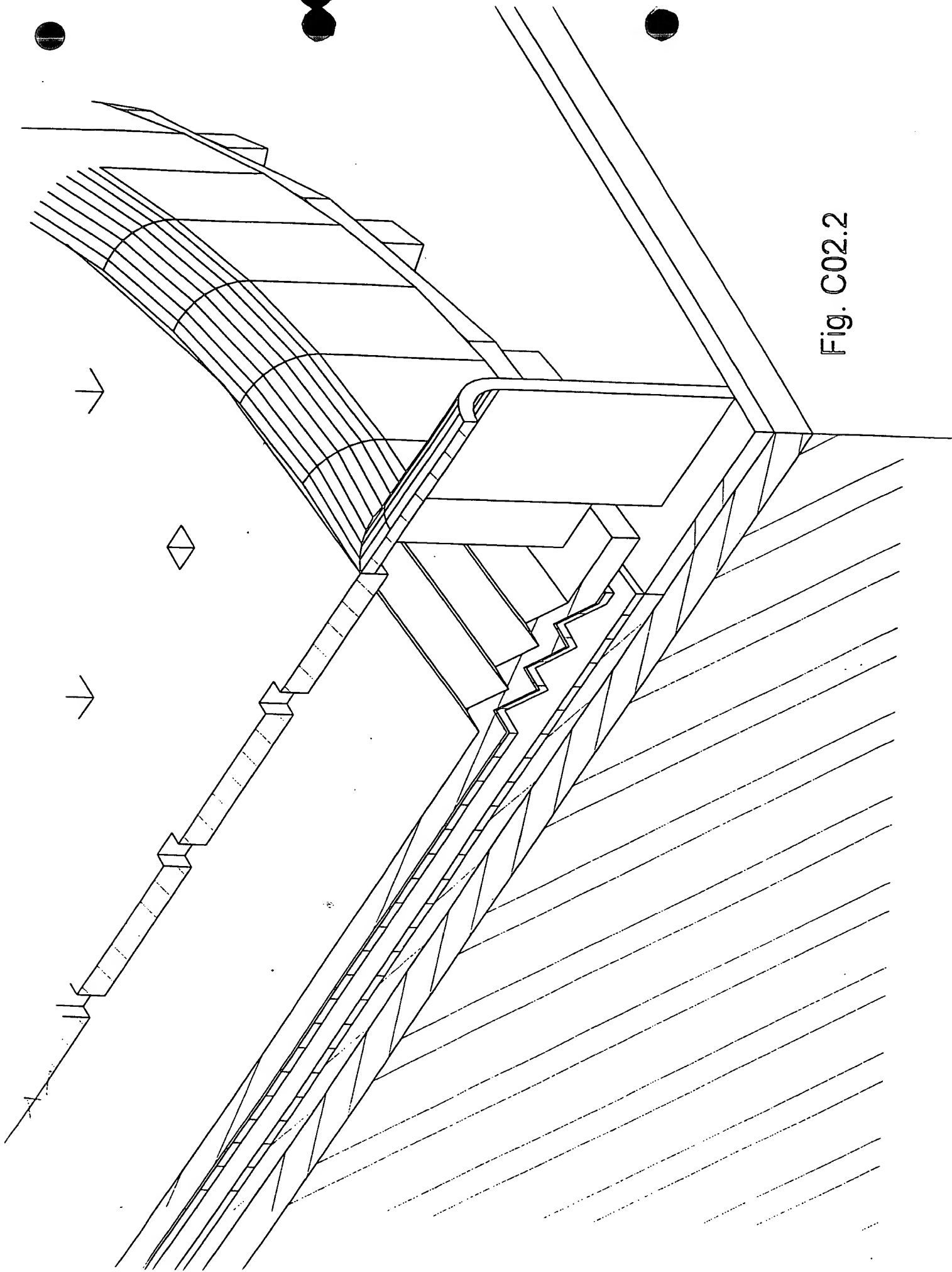


Fig. C02.2

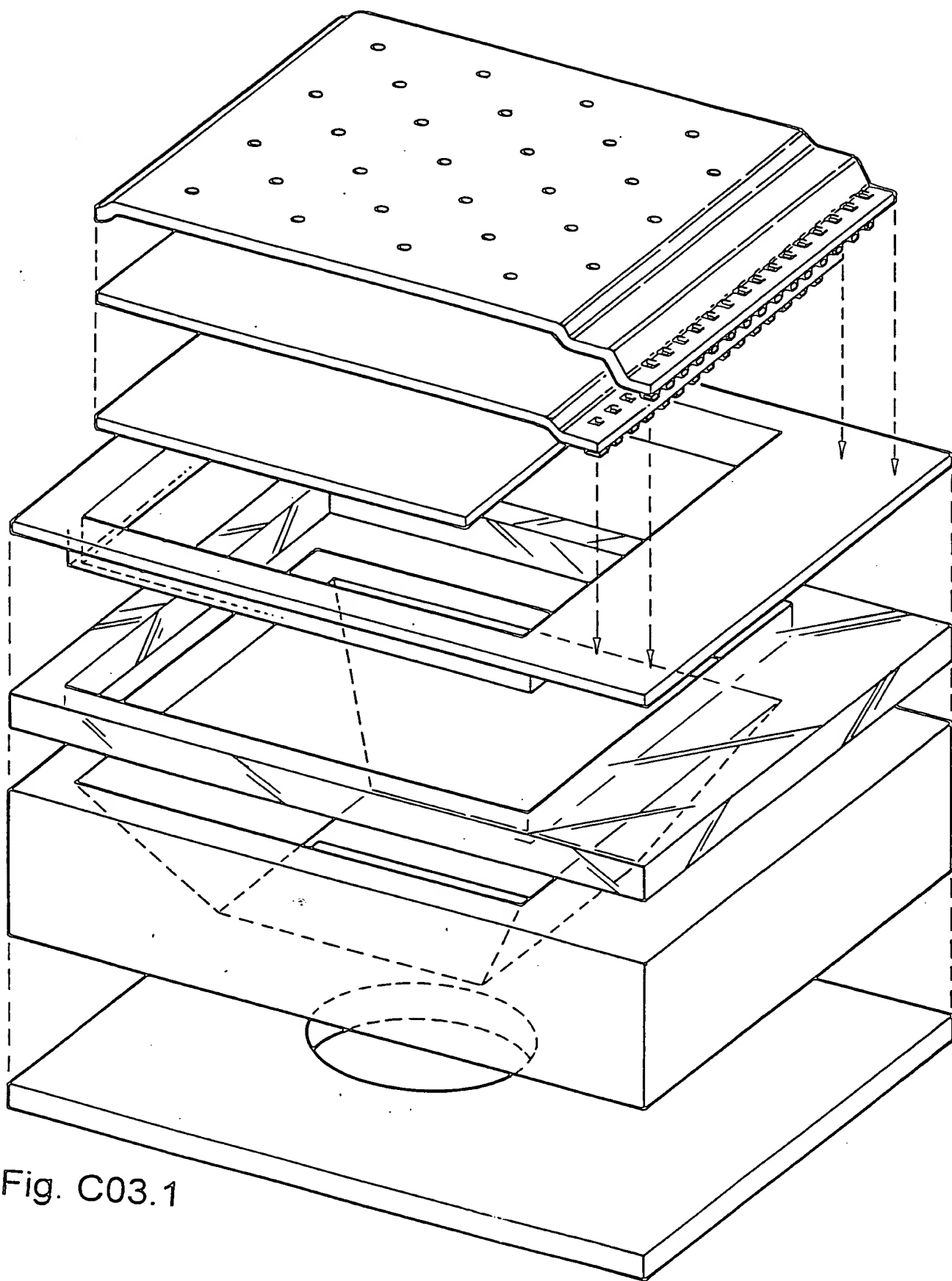


Fig. C03.1

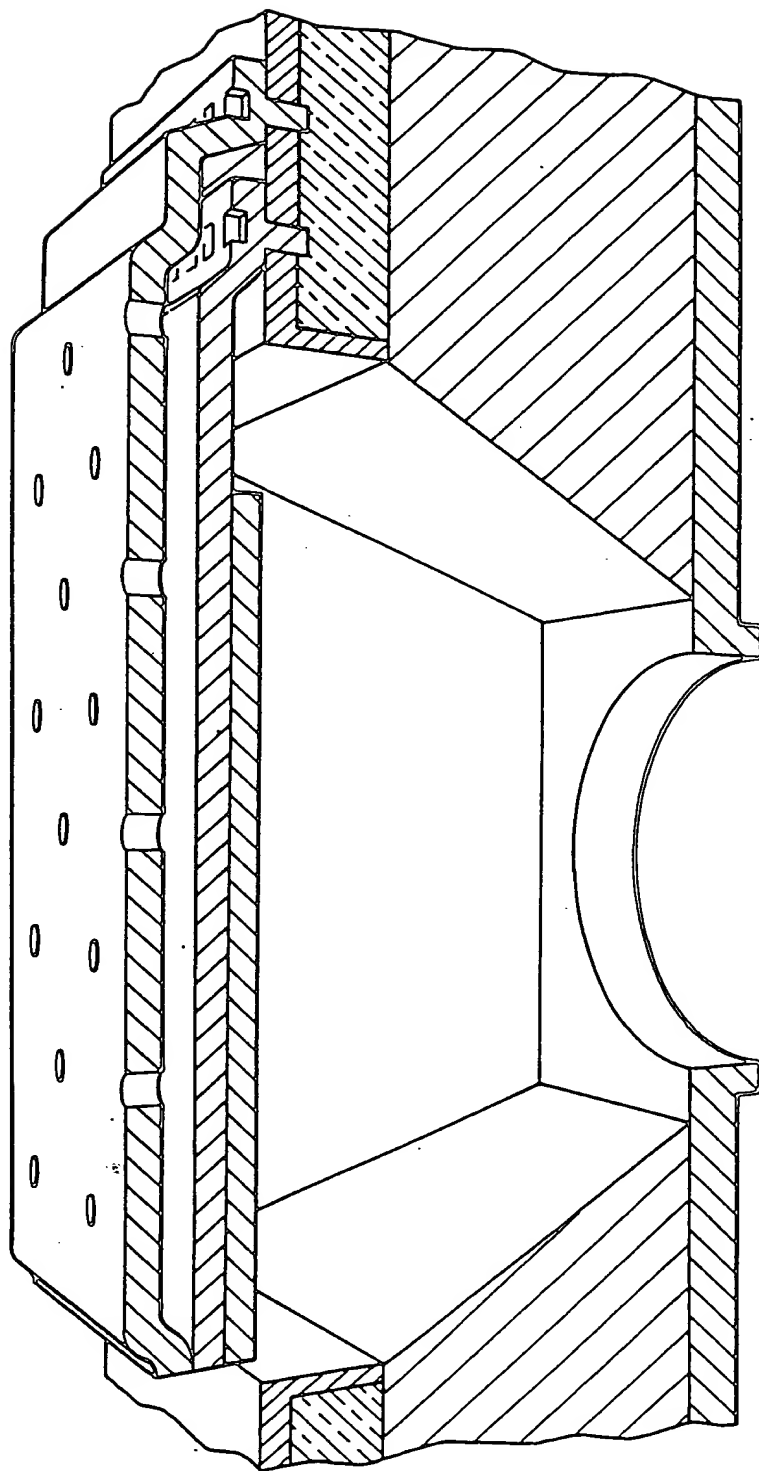


Fig. C03.2

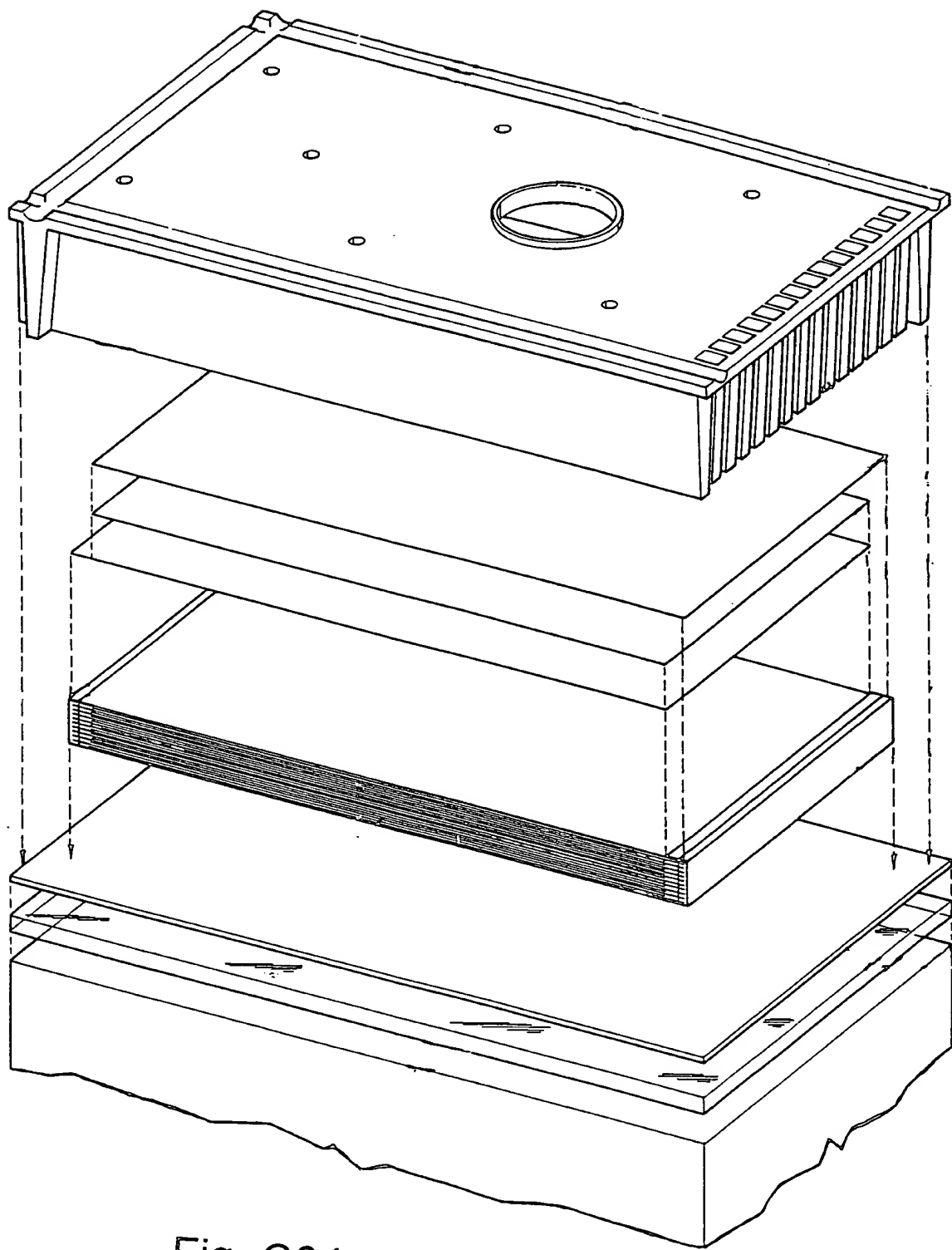


Fig. C04.1

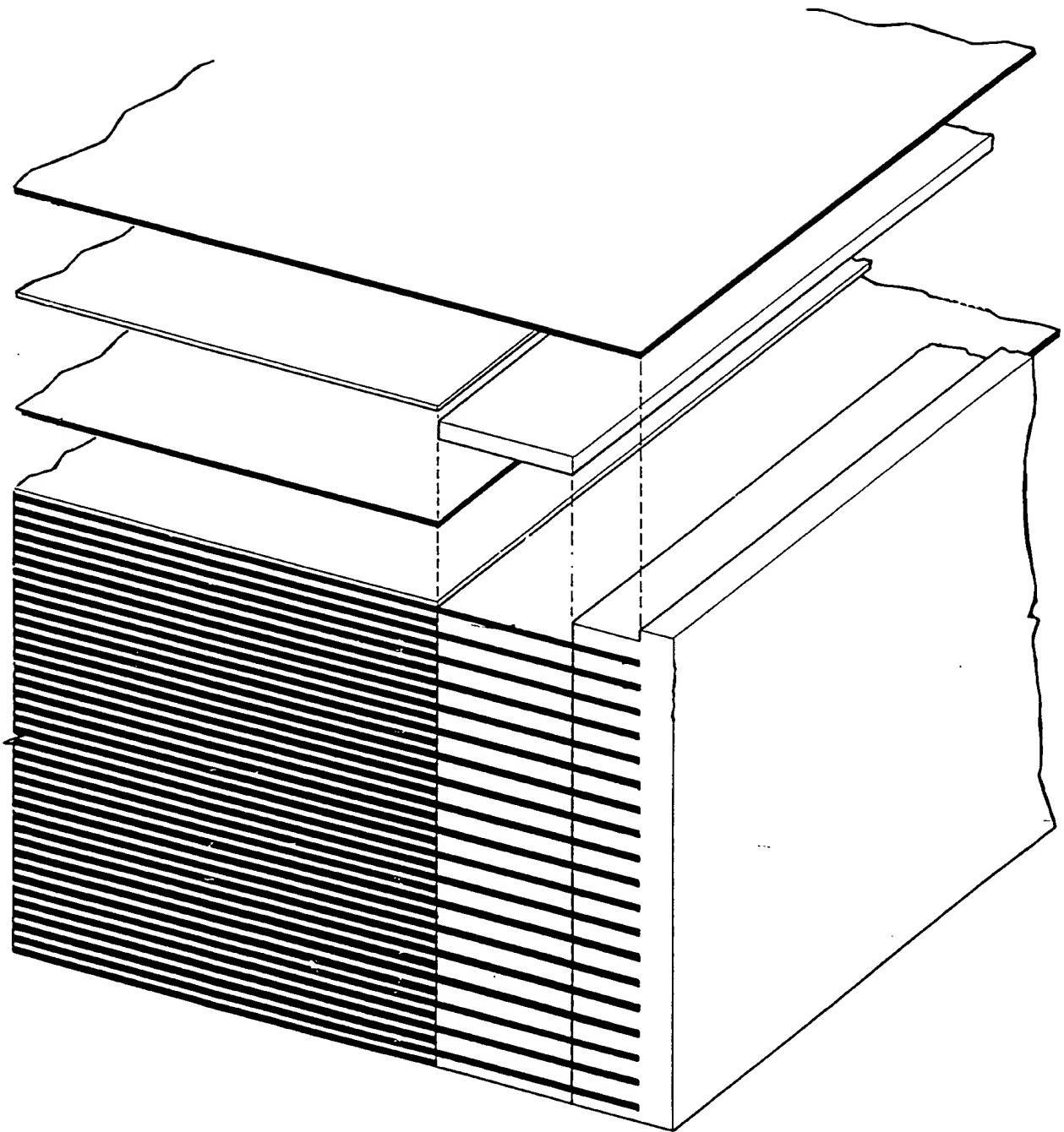


Fig. C04.2

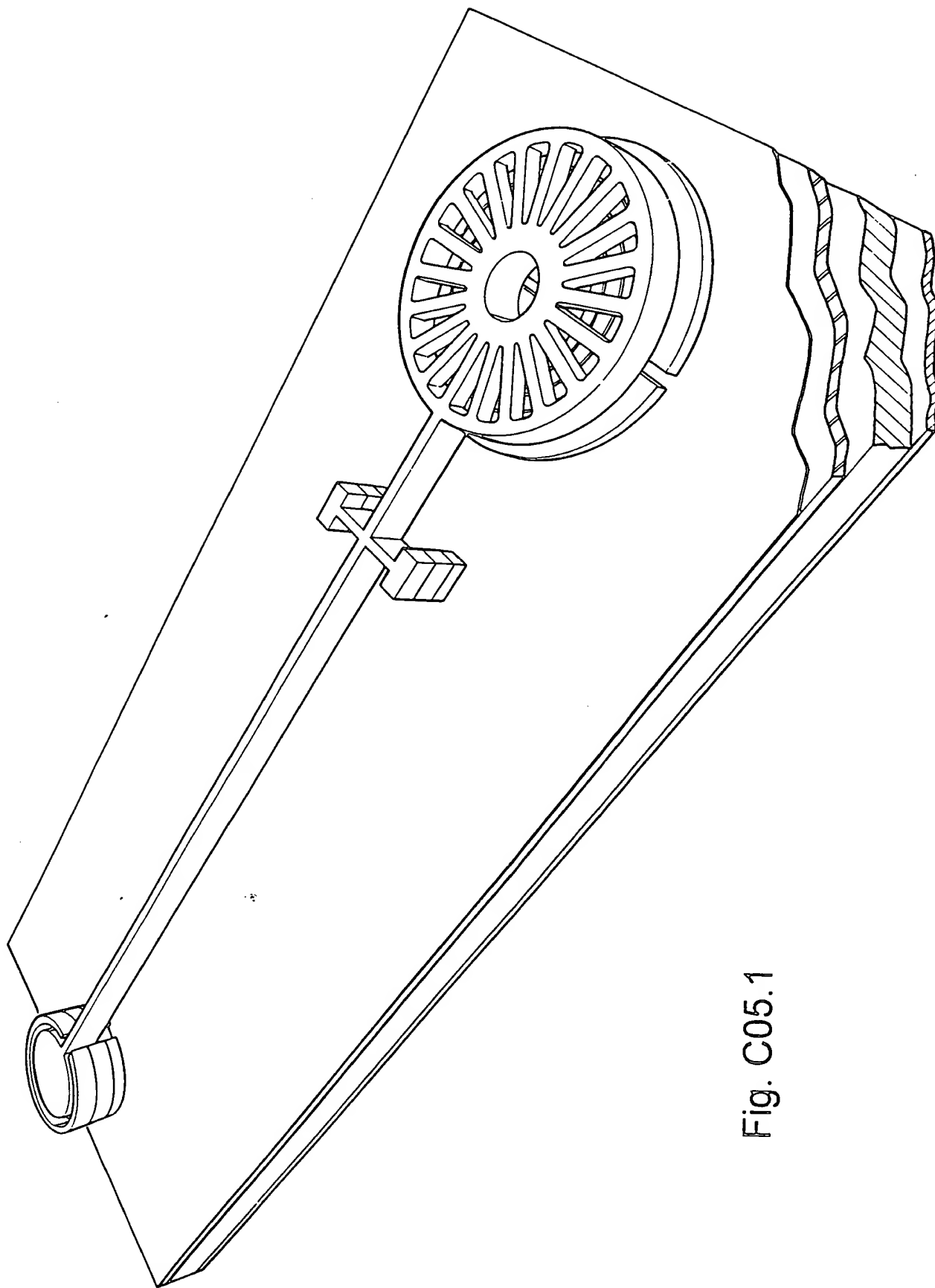


Fig. C05.1

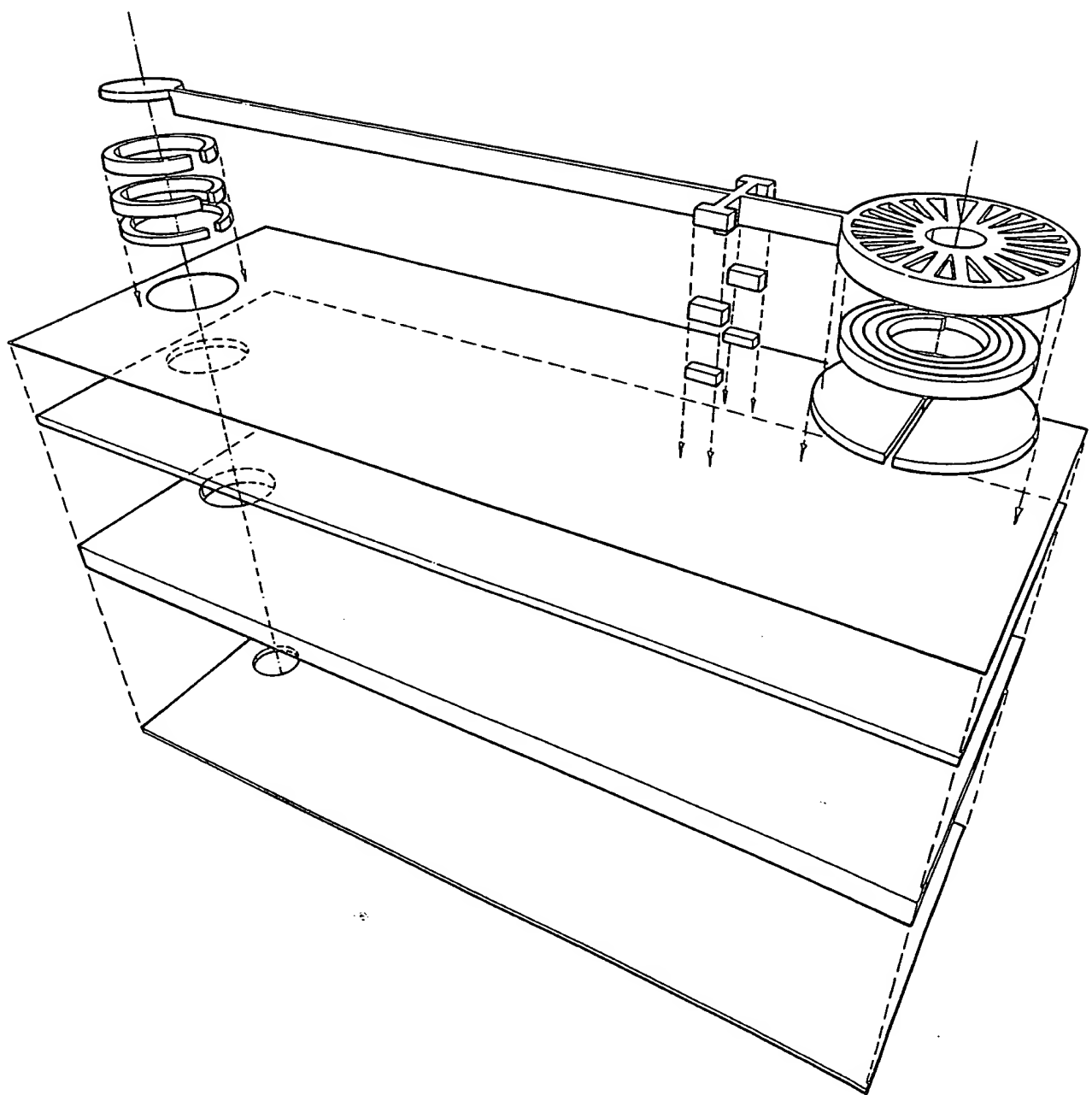


Fig. C05.2

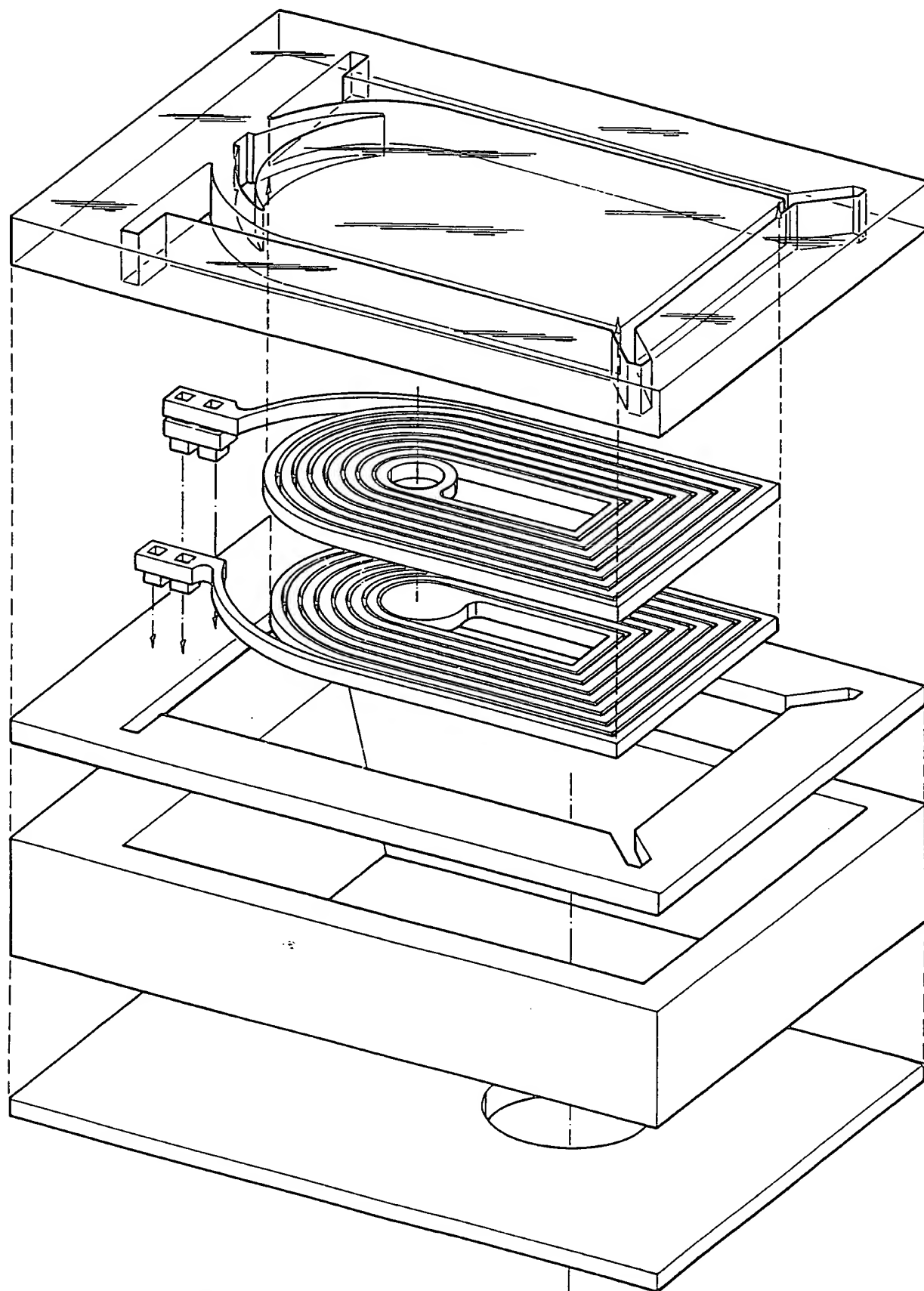


Fig. C06.1

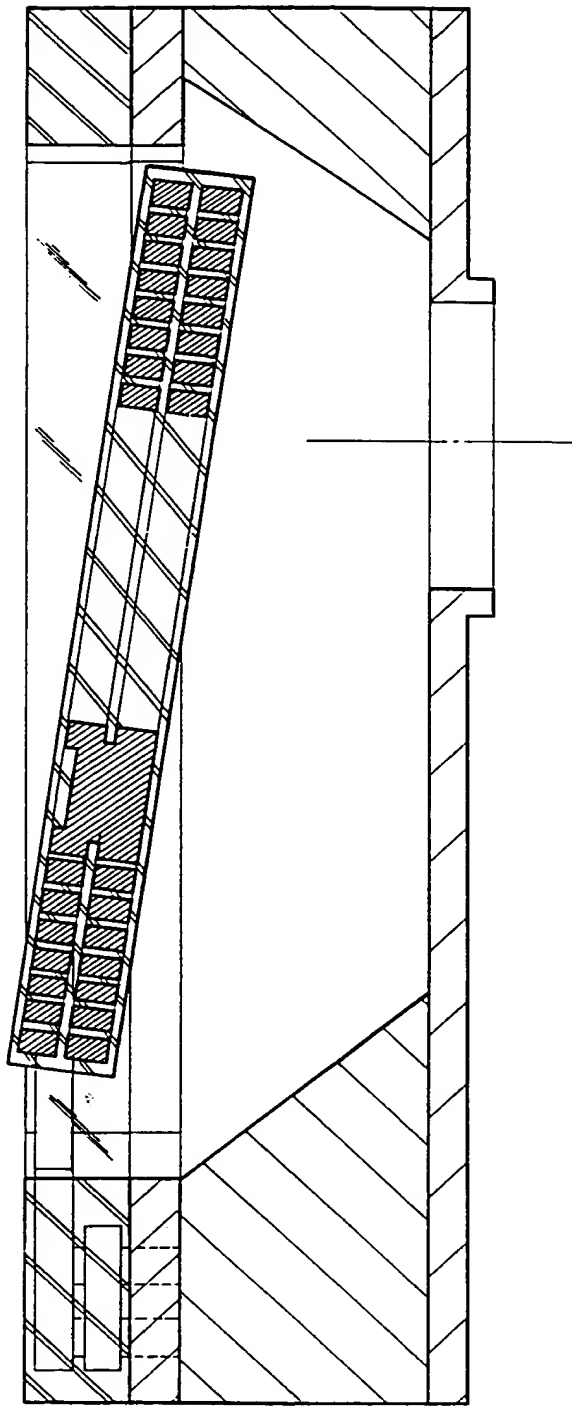


Fig. C06.2

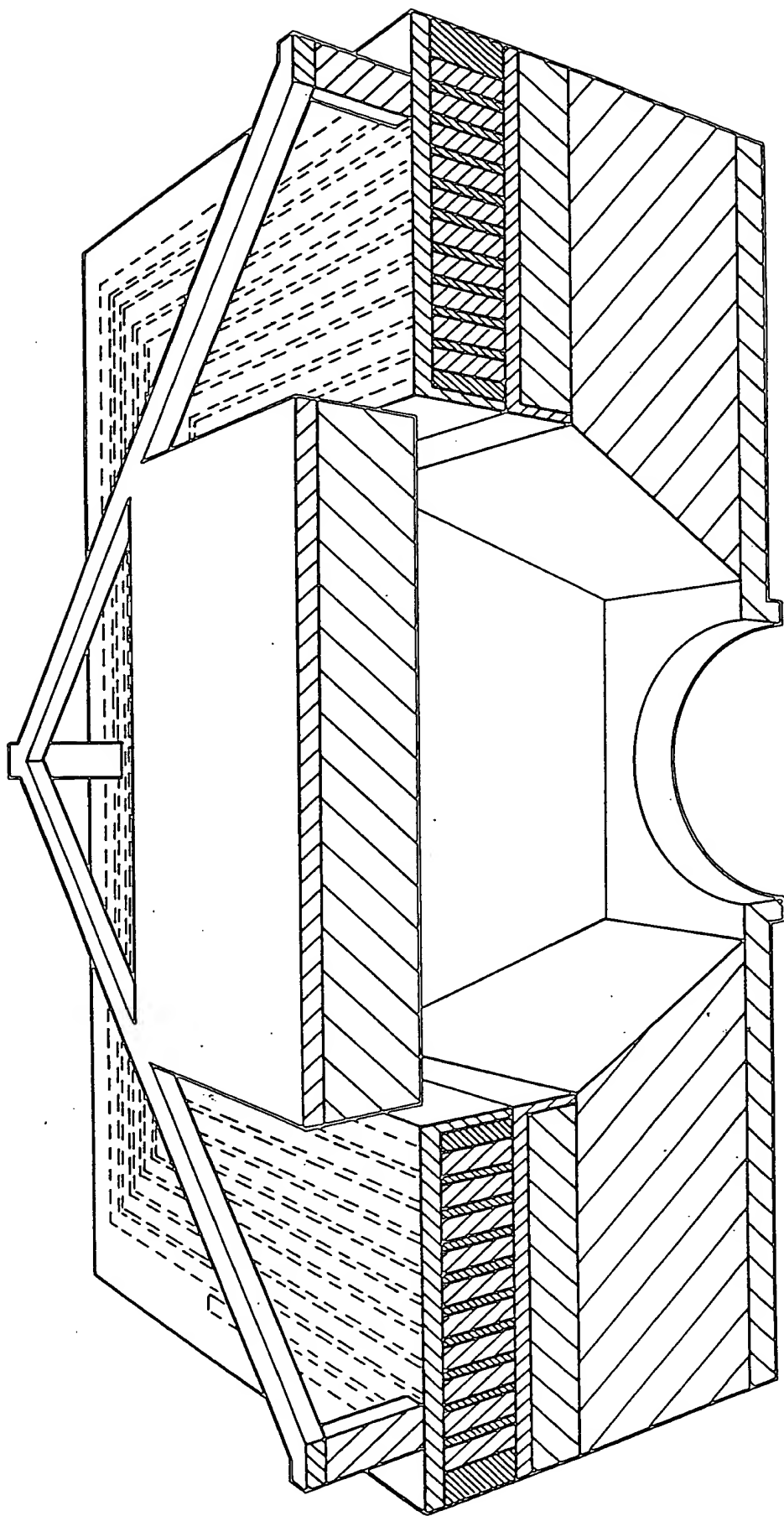


Fig. C07.1

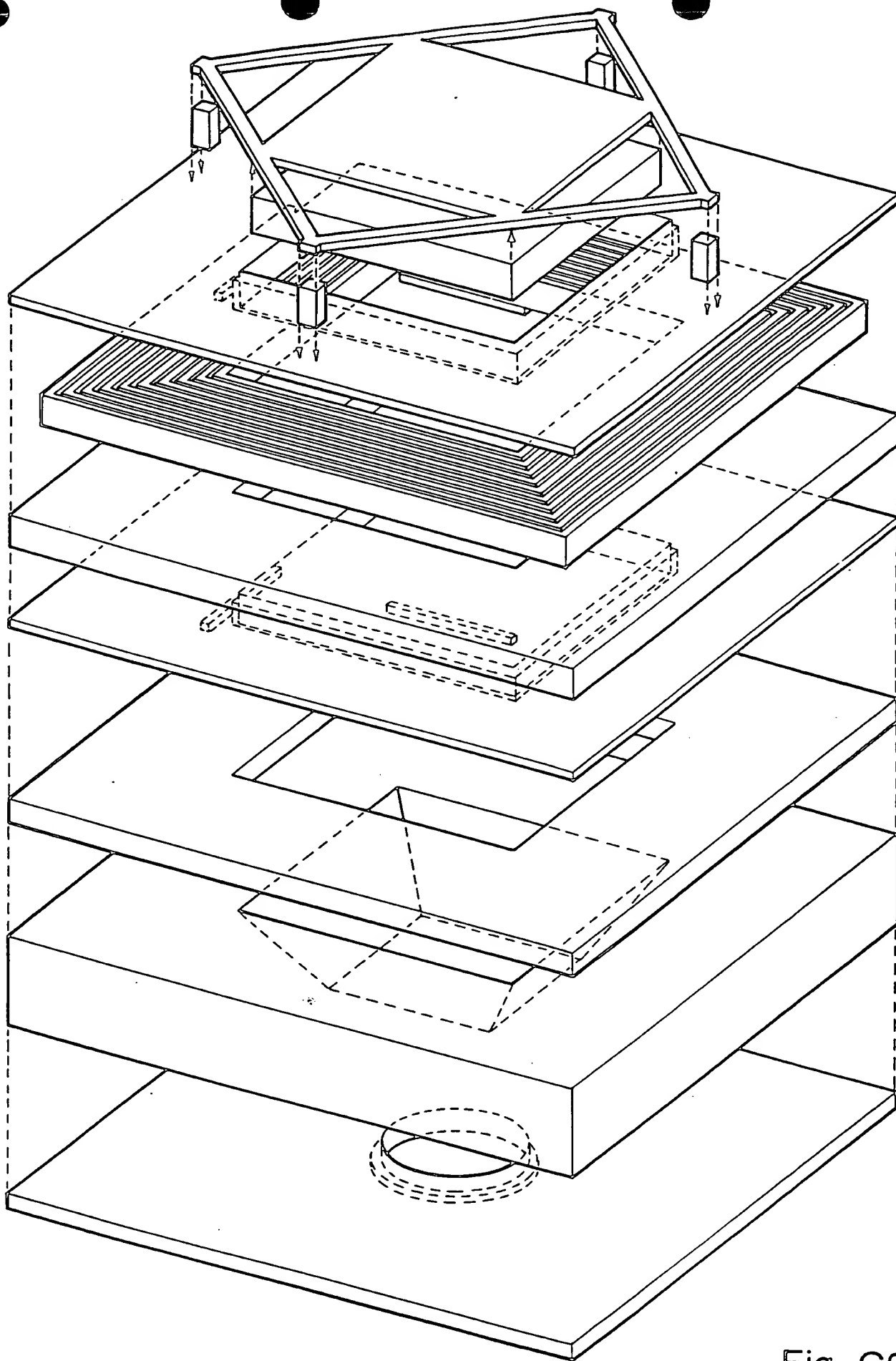


Fig. C07.2

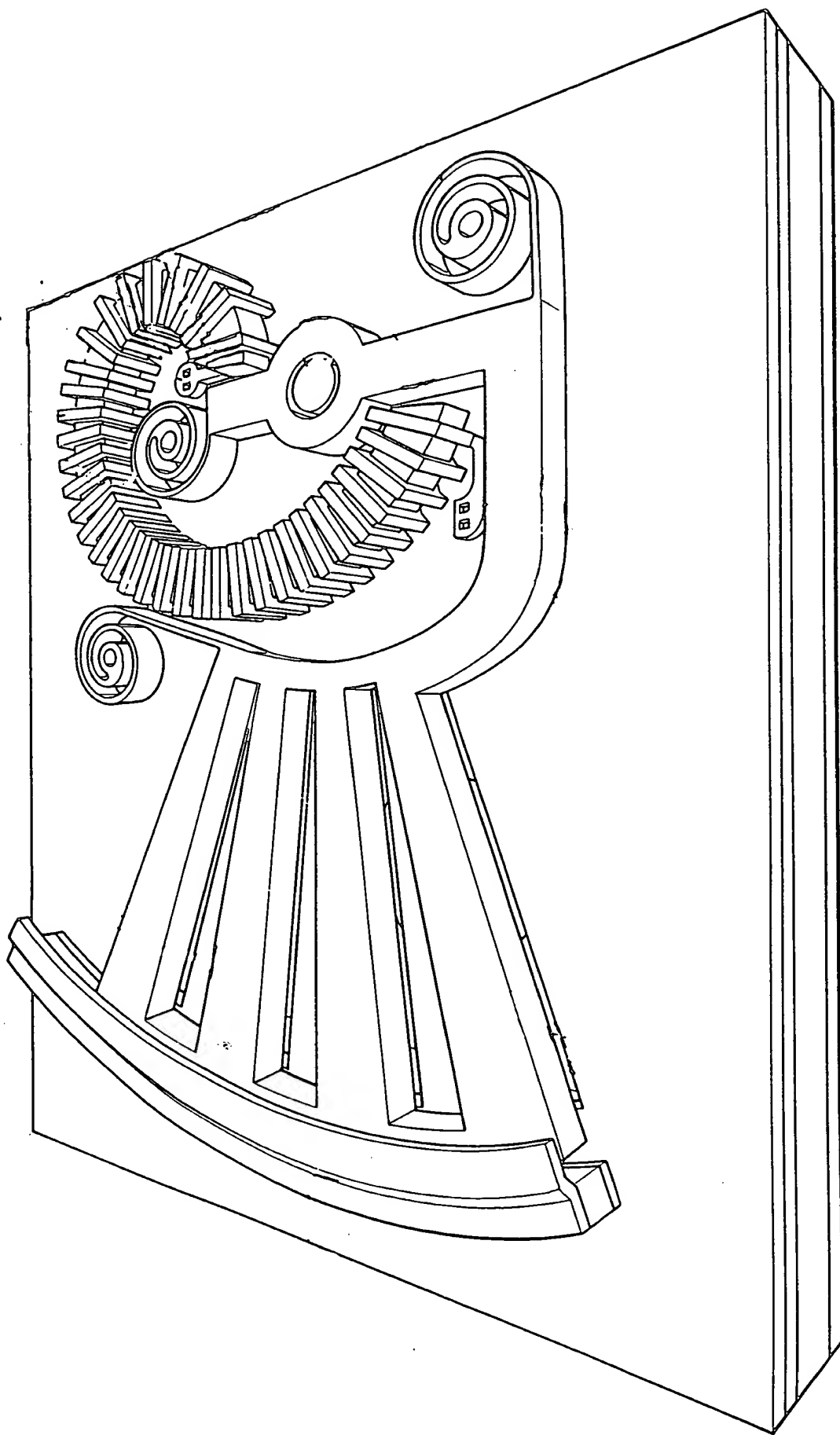


Fig. C08.1

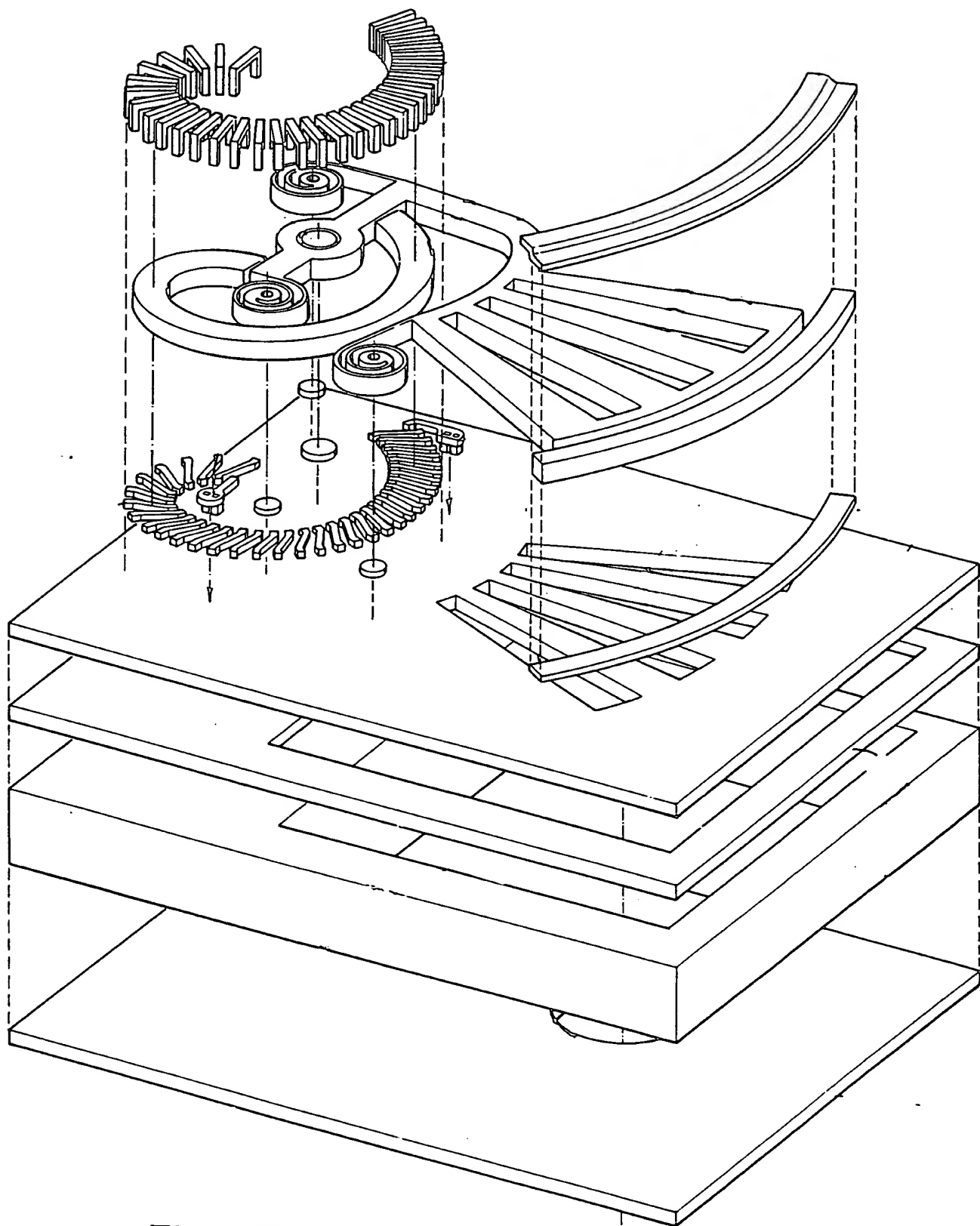


Fig. C08.2

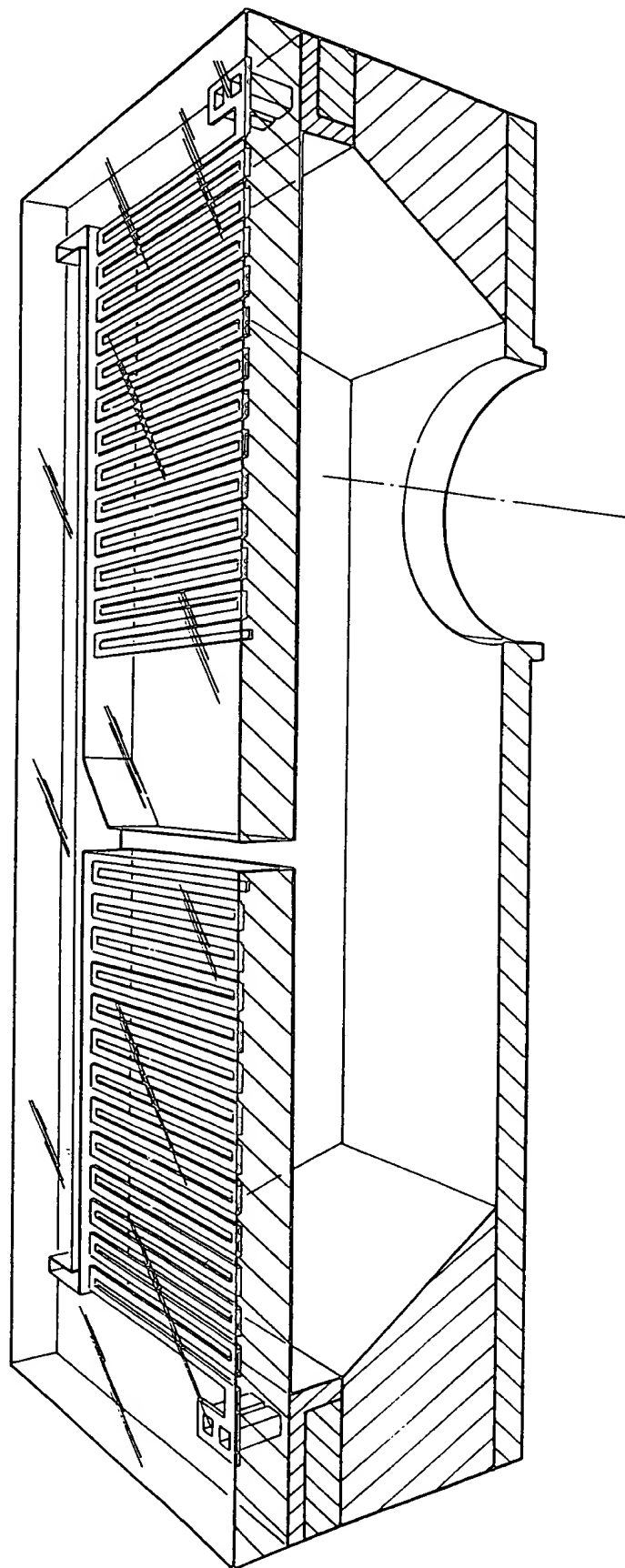


Fig. C09.1

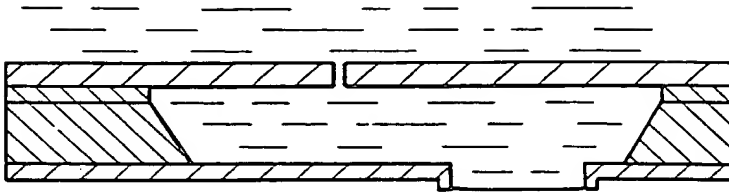


Fig. C09.2

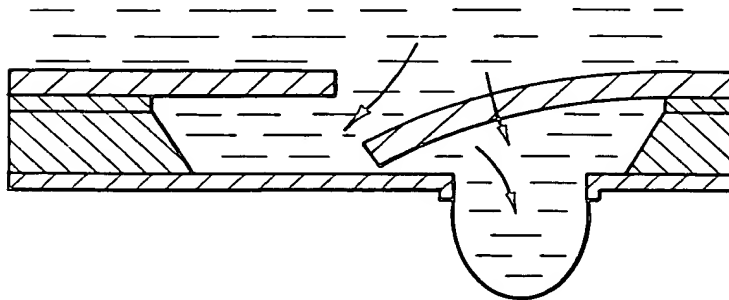


Fig. C09.3

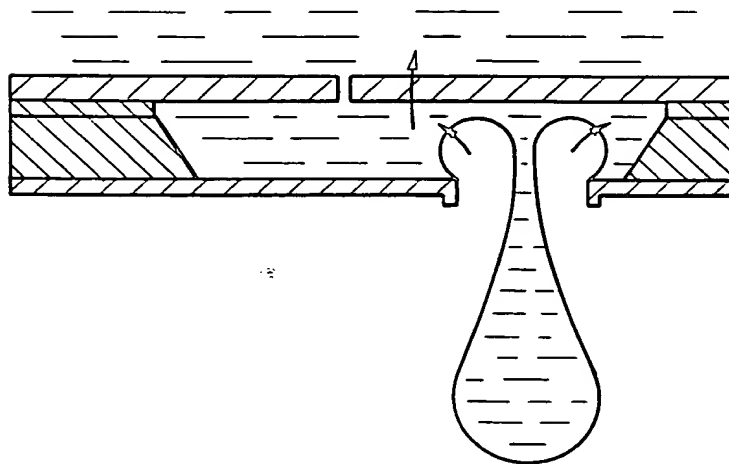


Fig. C09.4

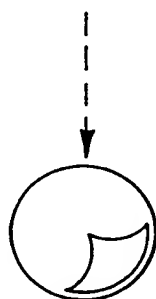
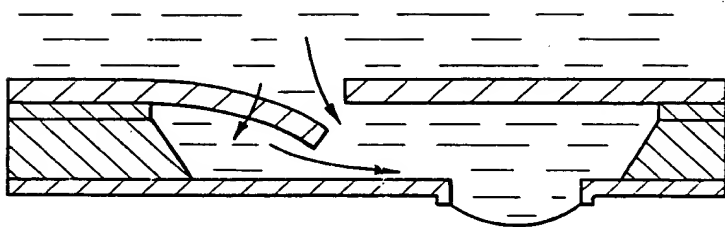


Fig. C09.5

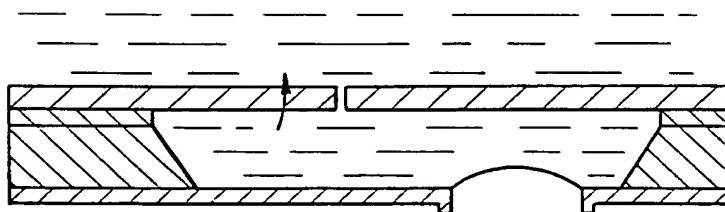


Fig. C09.6

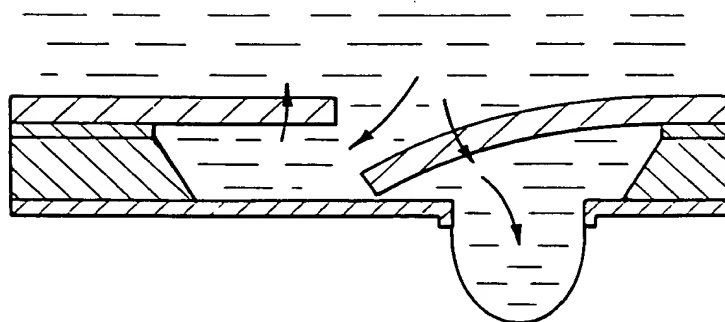


Fig. C09.7

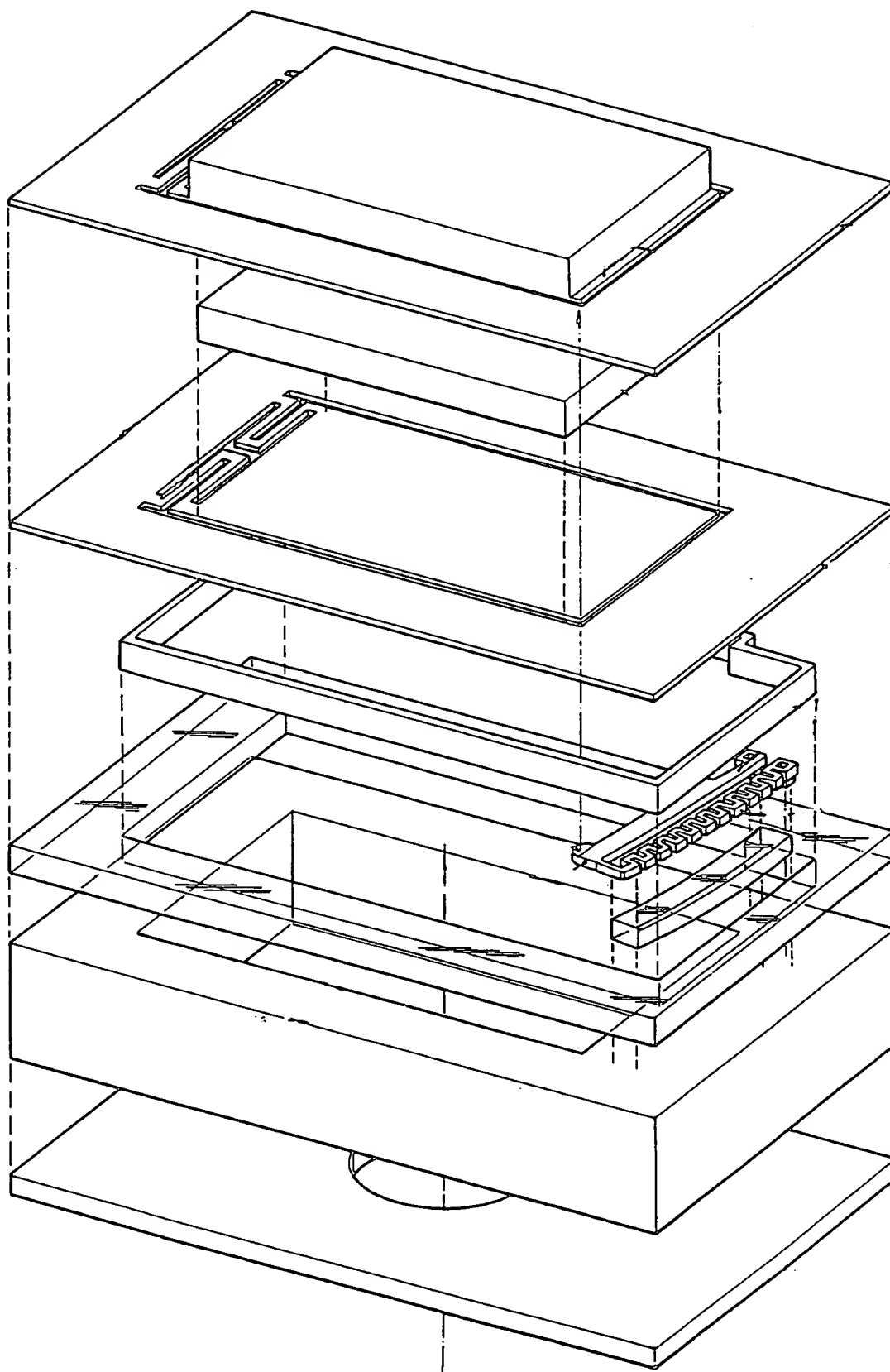


Fig. C10.1

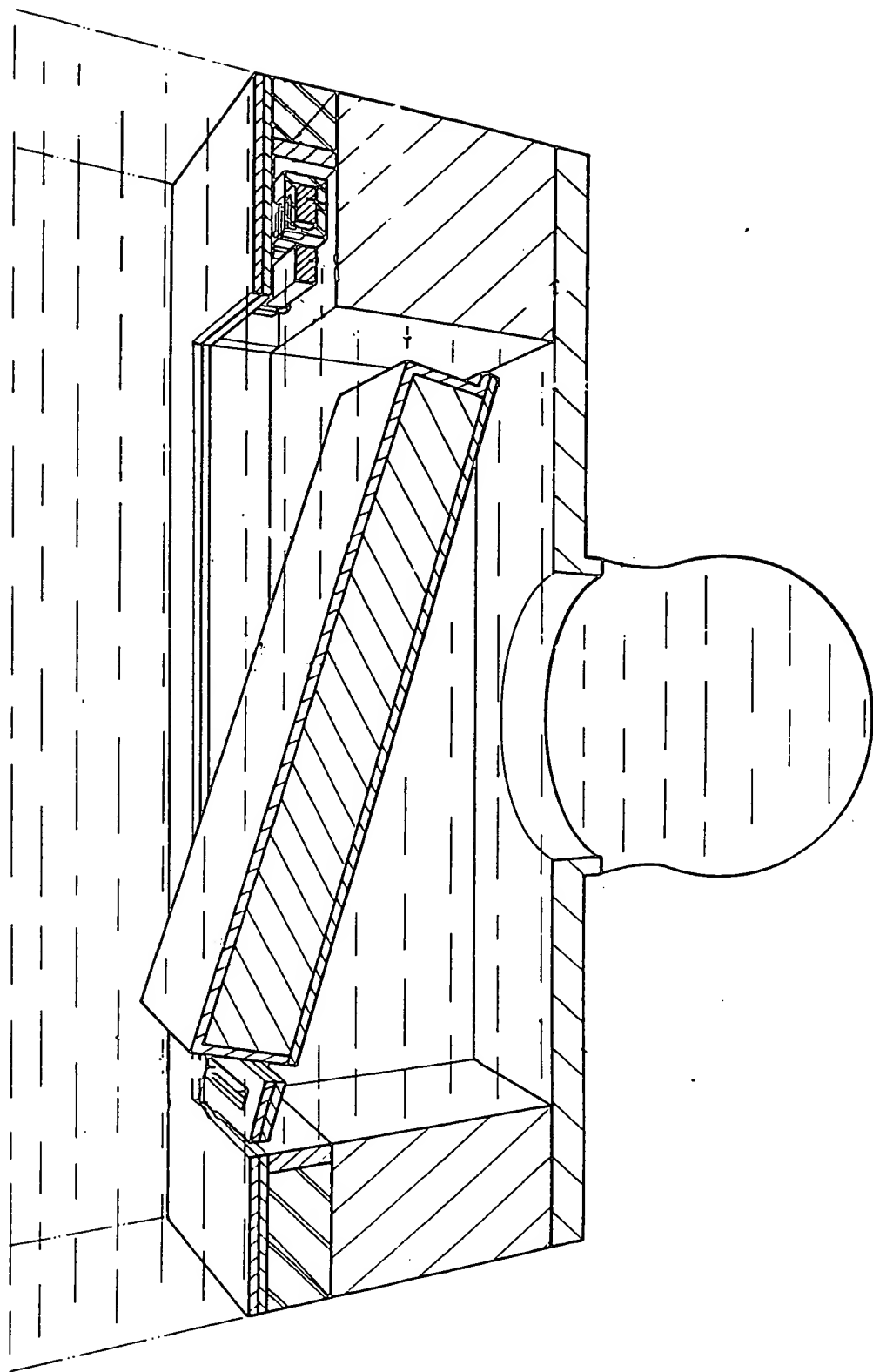


Fig. C10.2

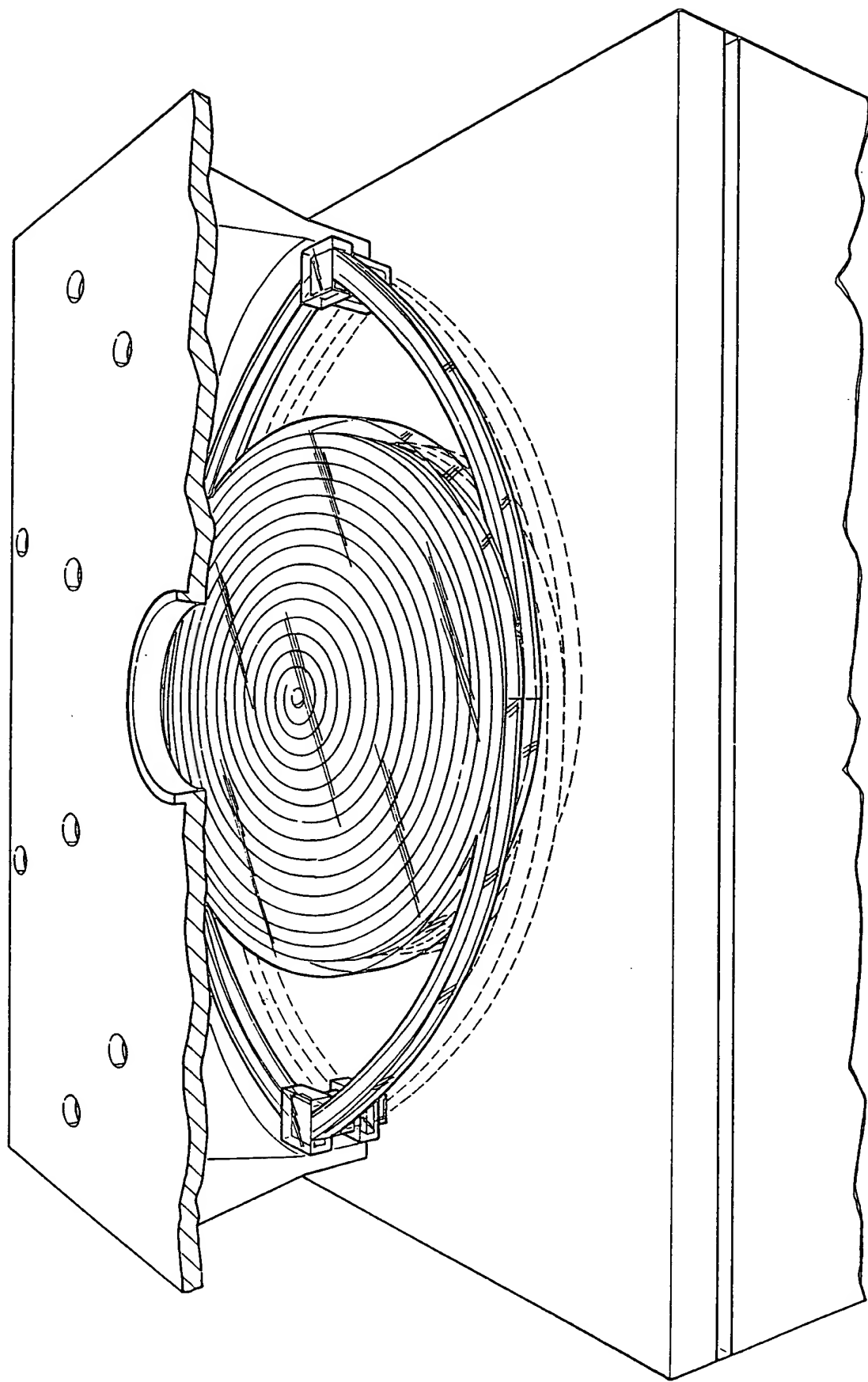


Fig. C11.1

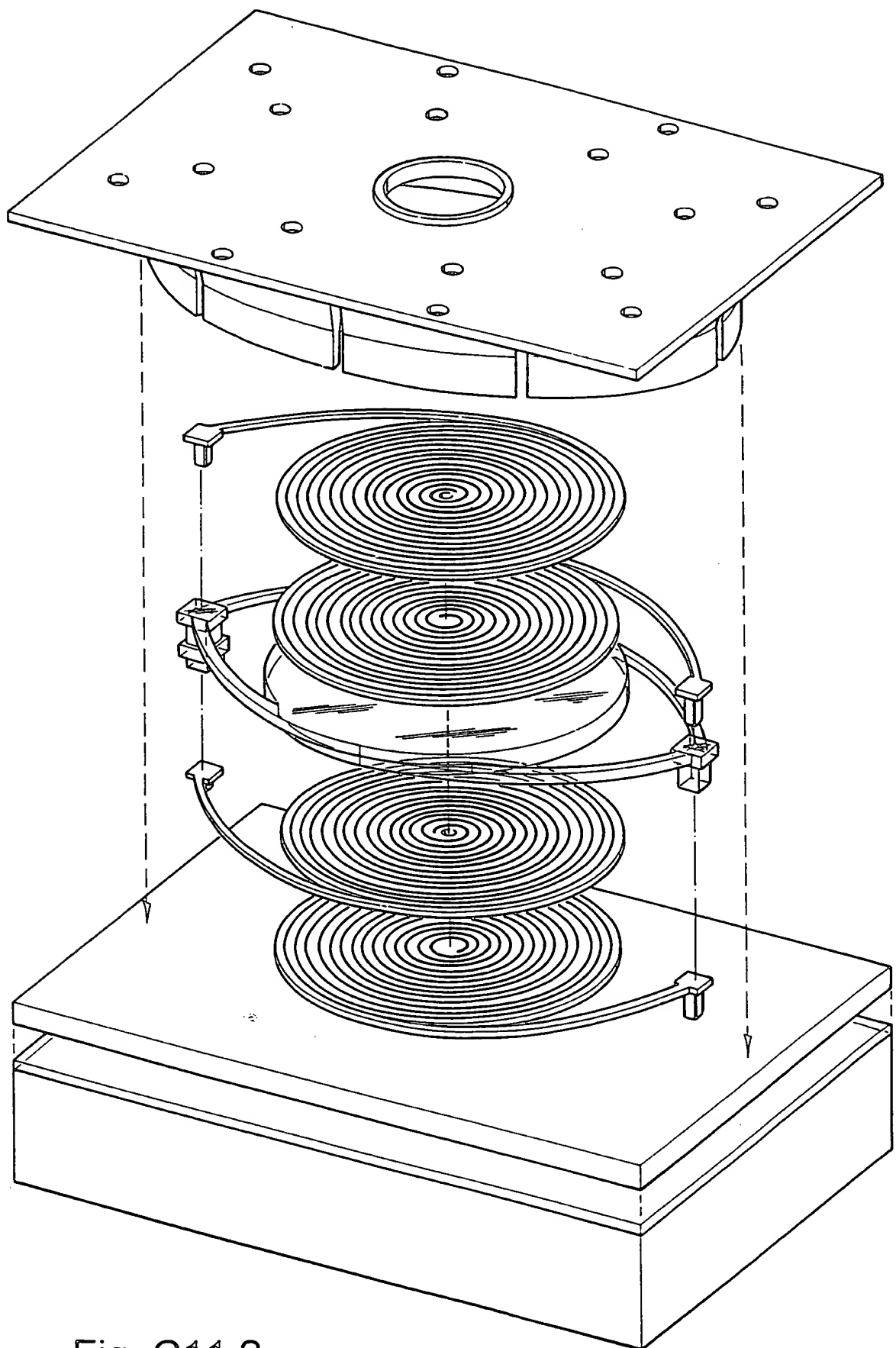


Fig. C11.2

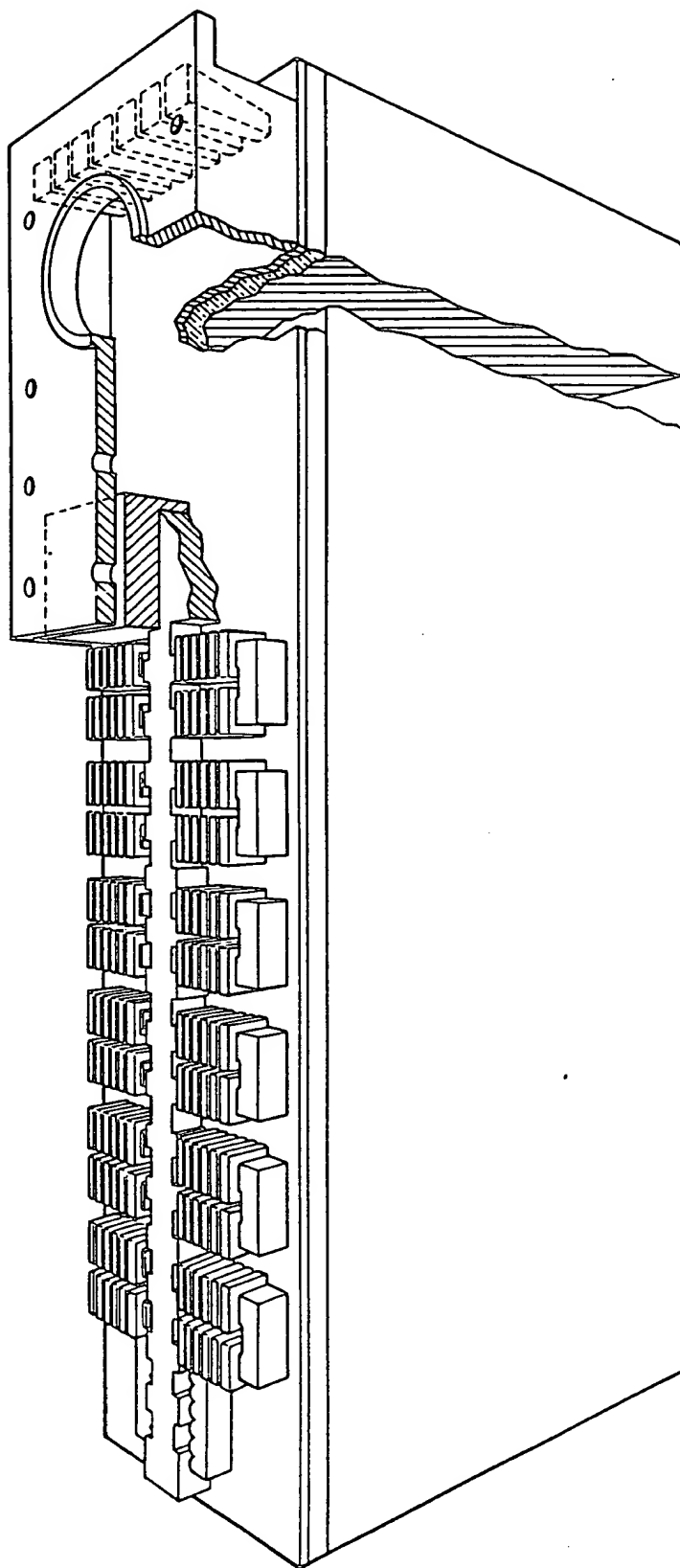


Fig. C12.1

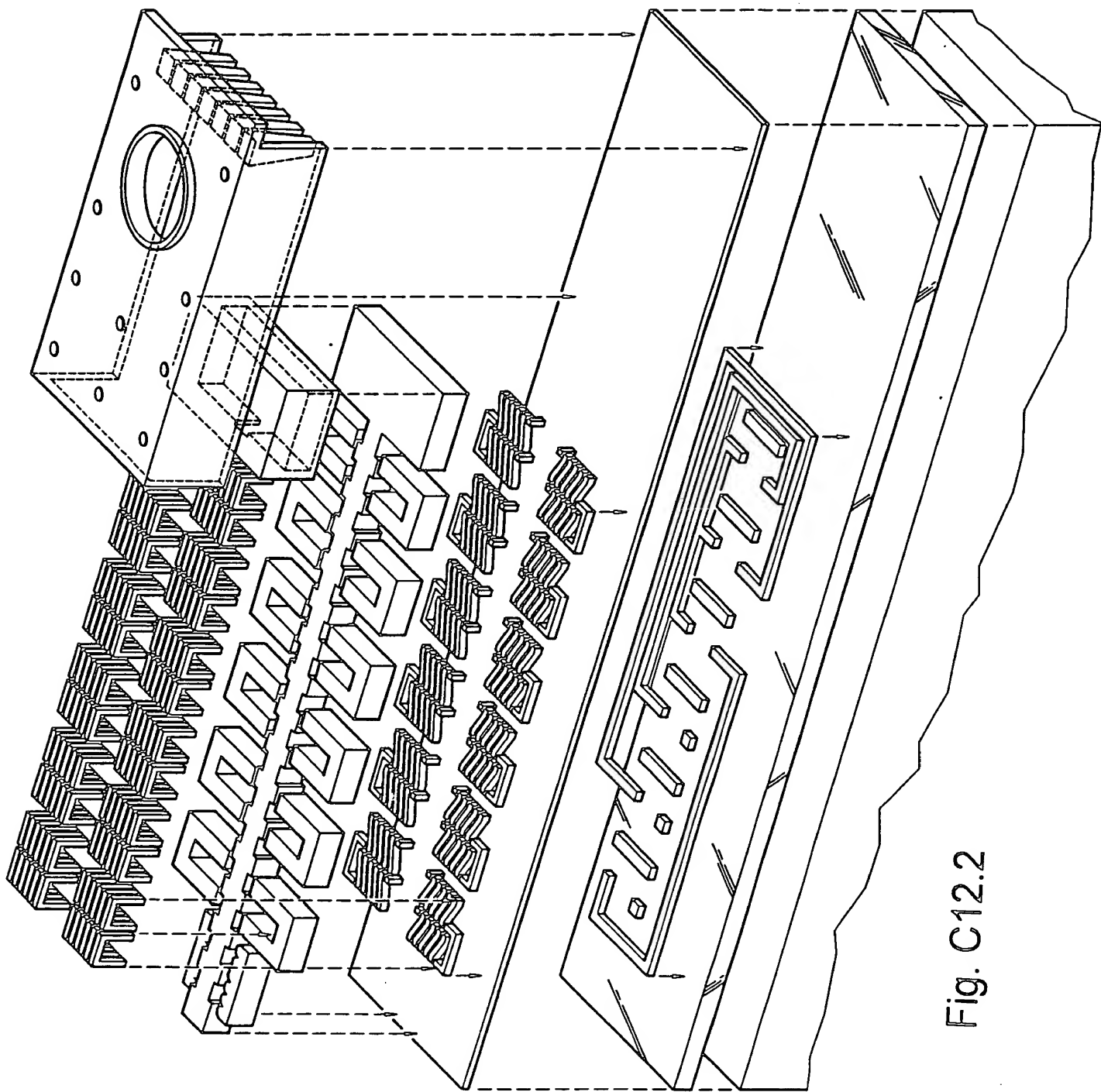


Fig. C12.2

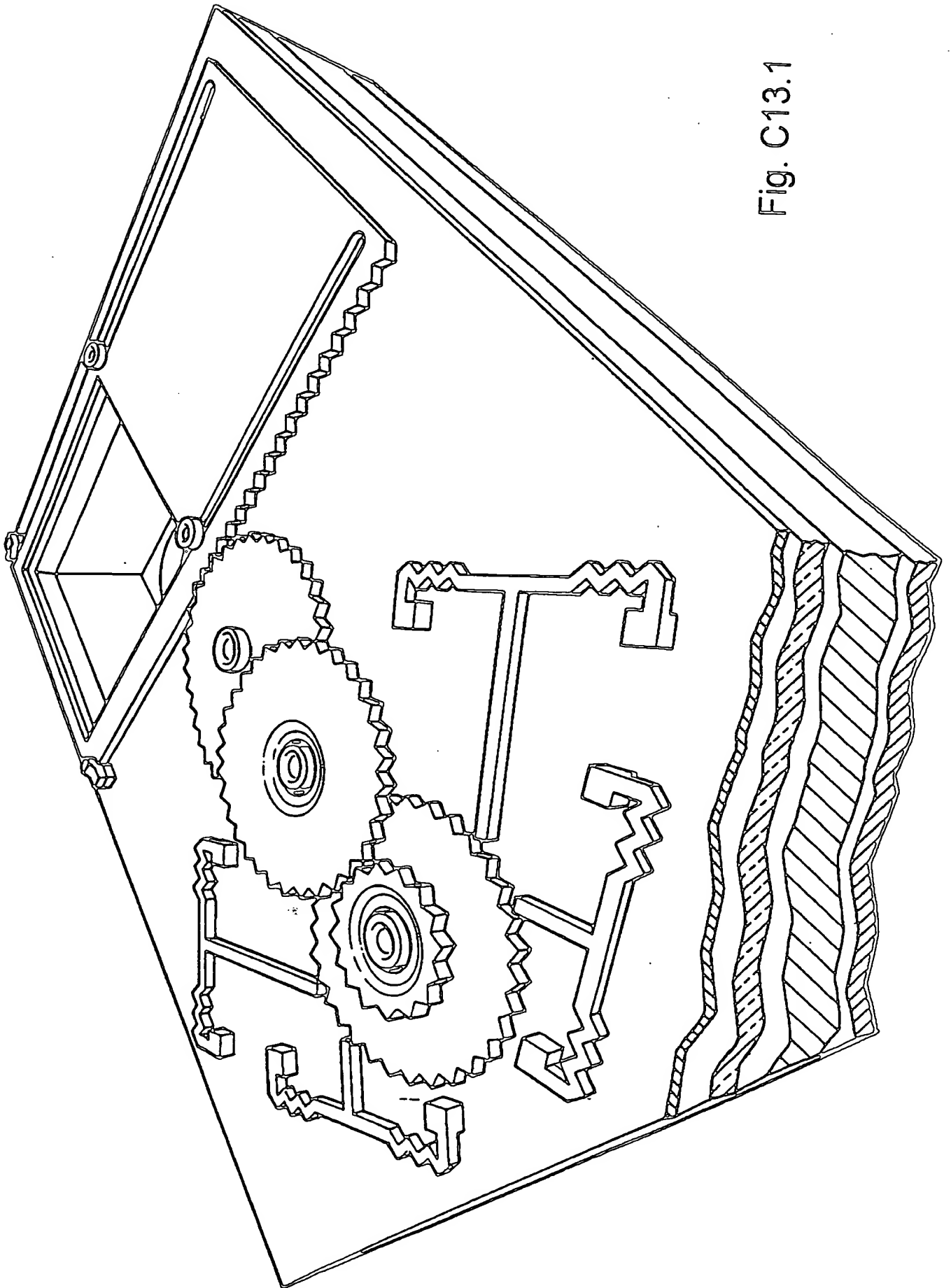


Fig. C13.1

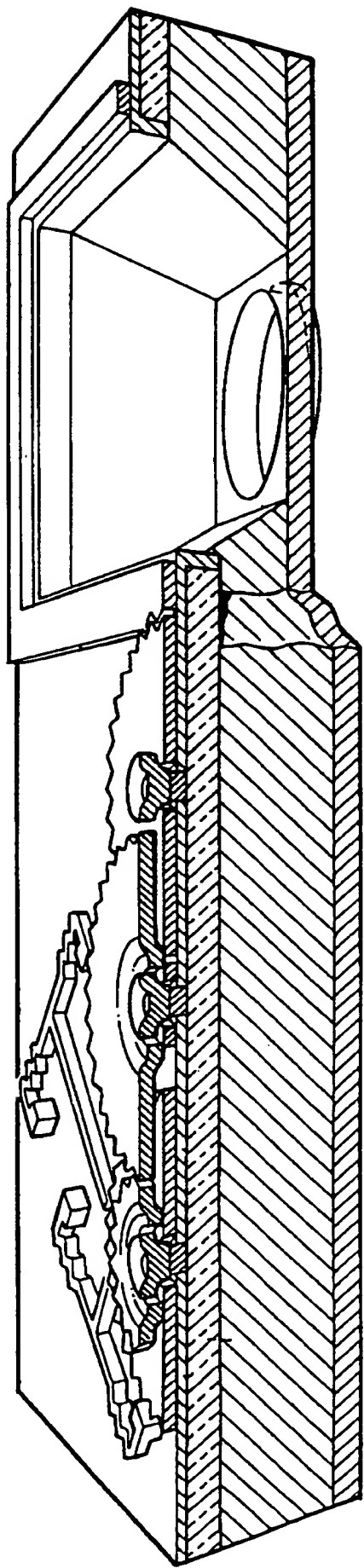


Fig. C13.2

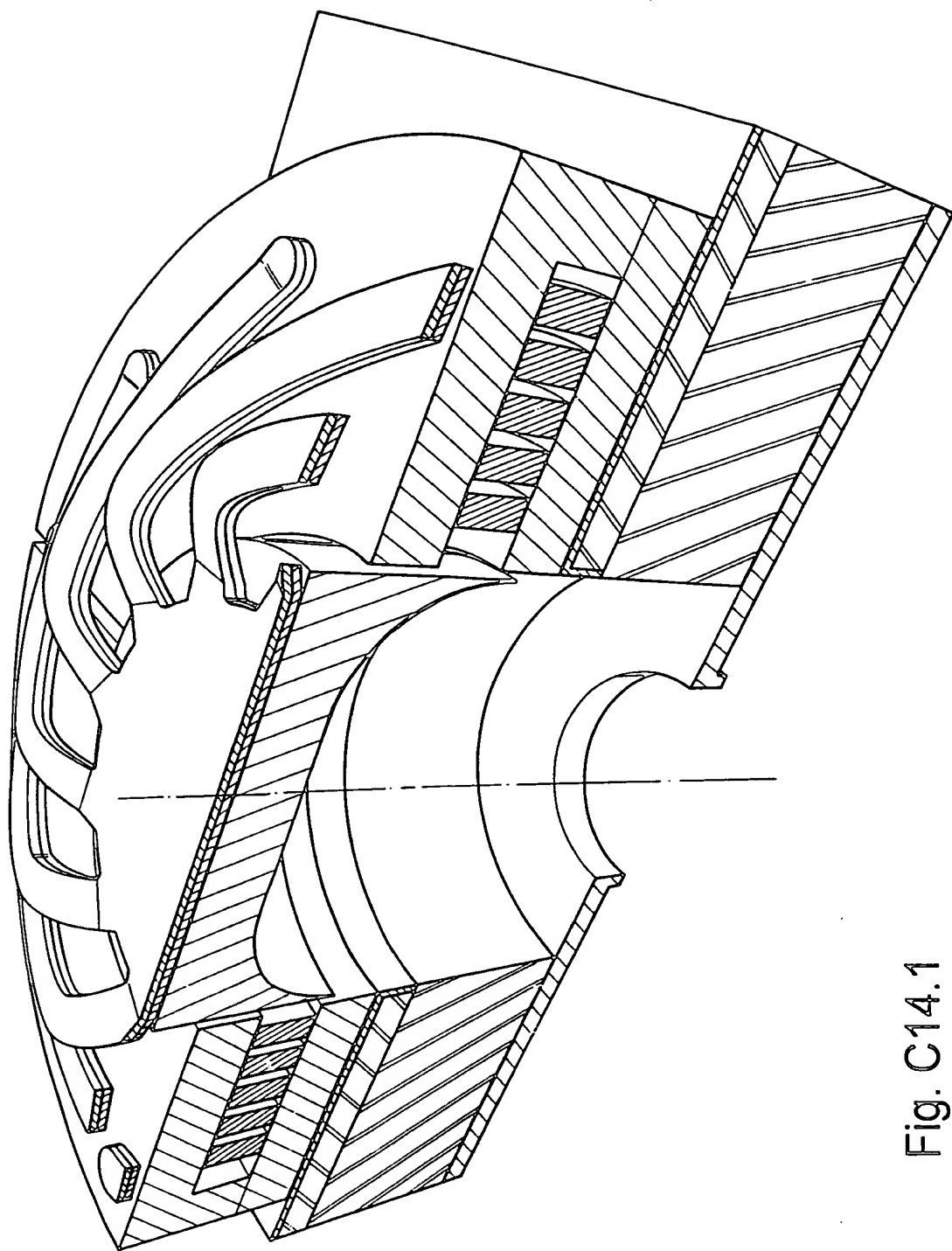


Fig. C14.1

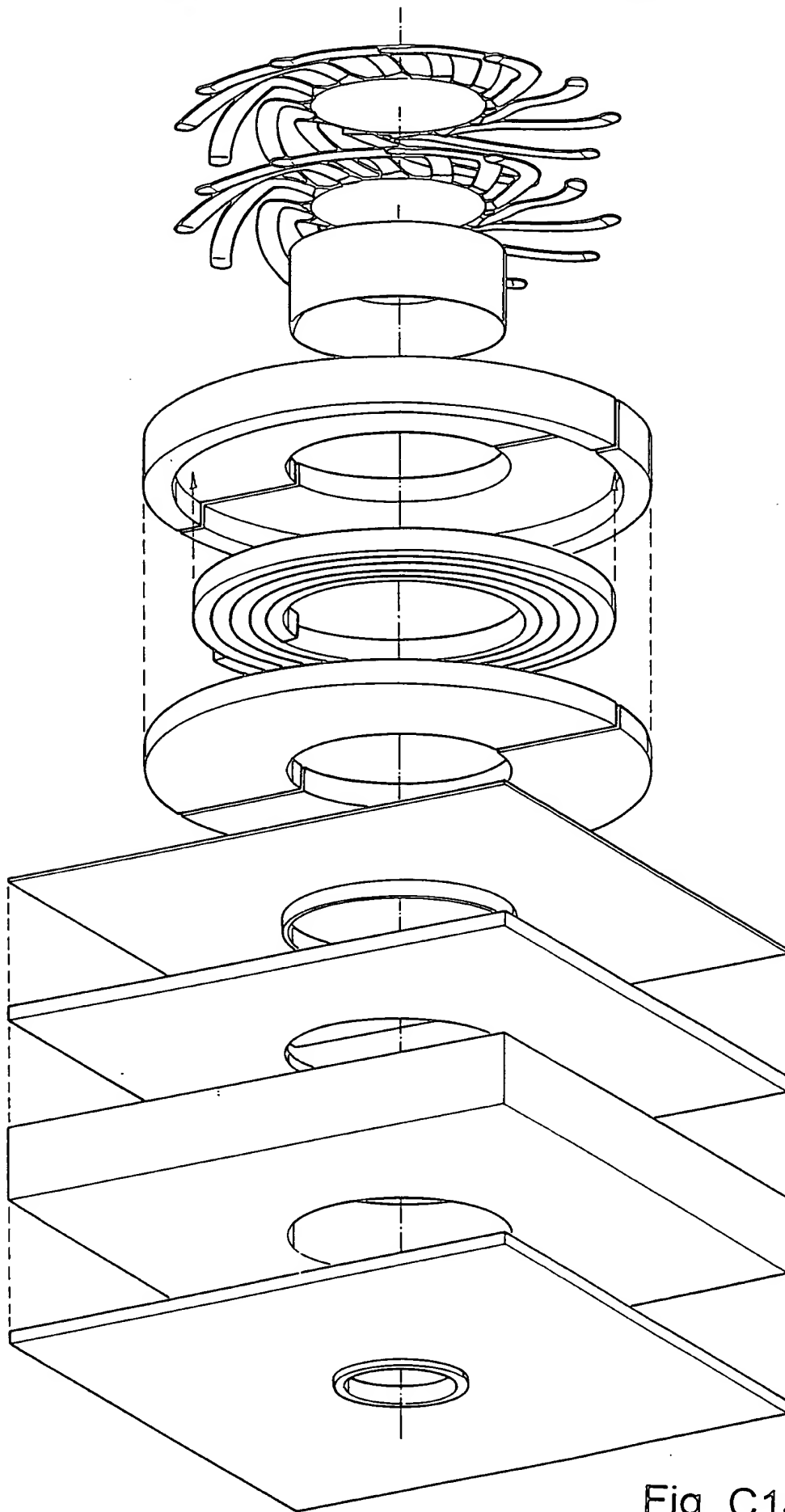


Fig. C14.2

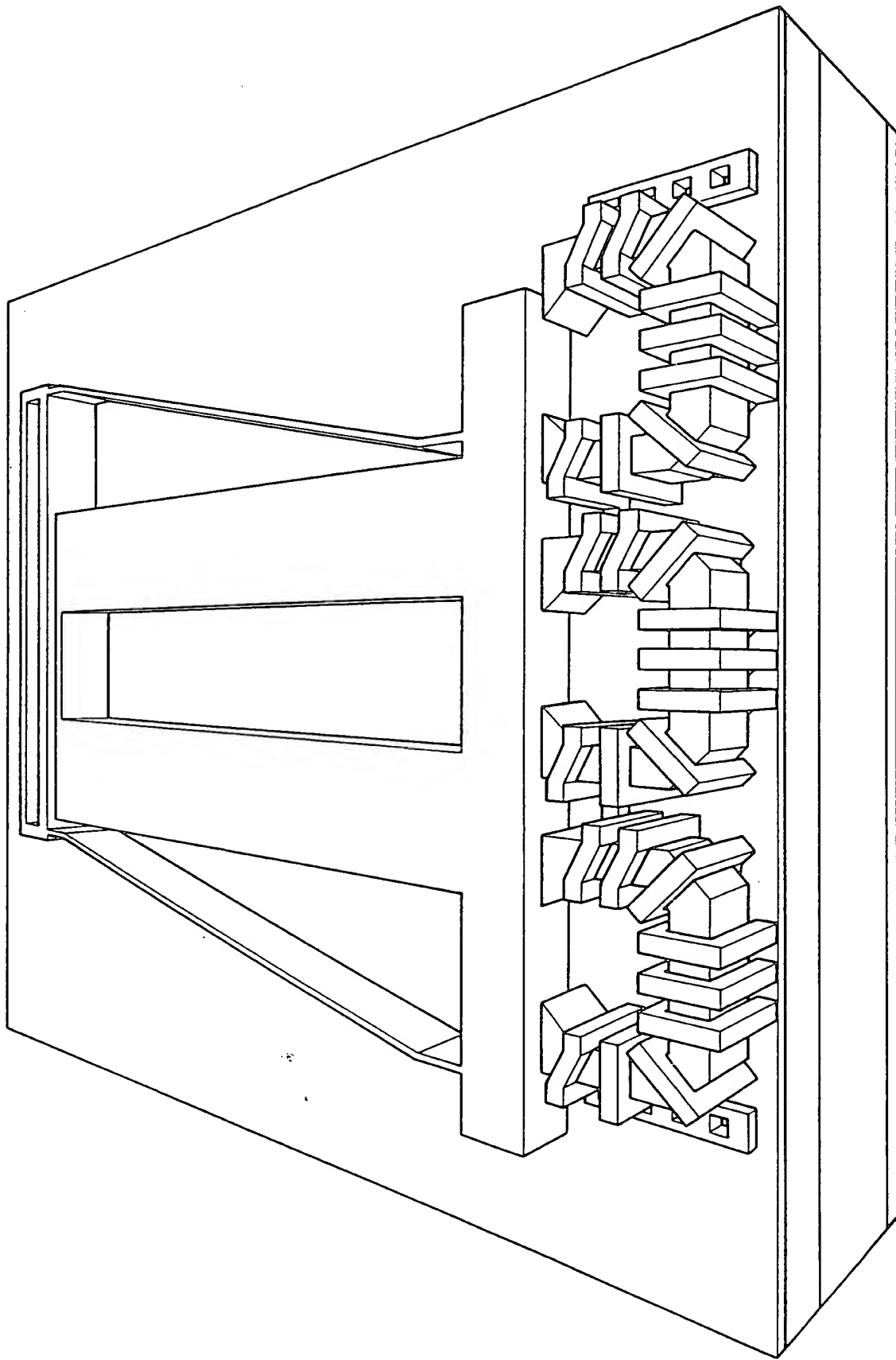


Fig. C15.1

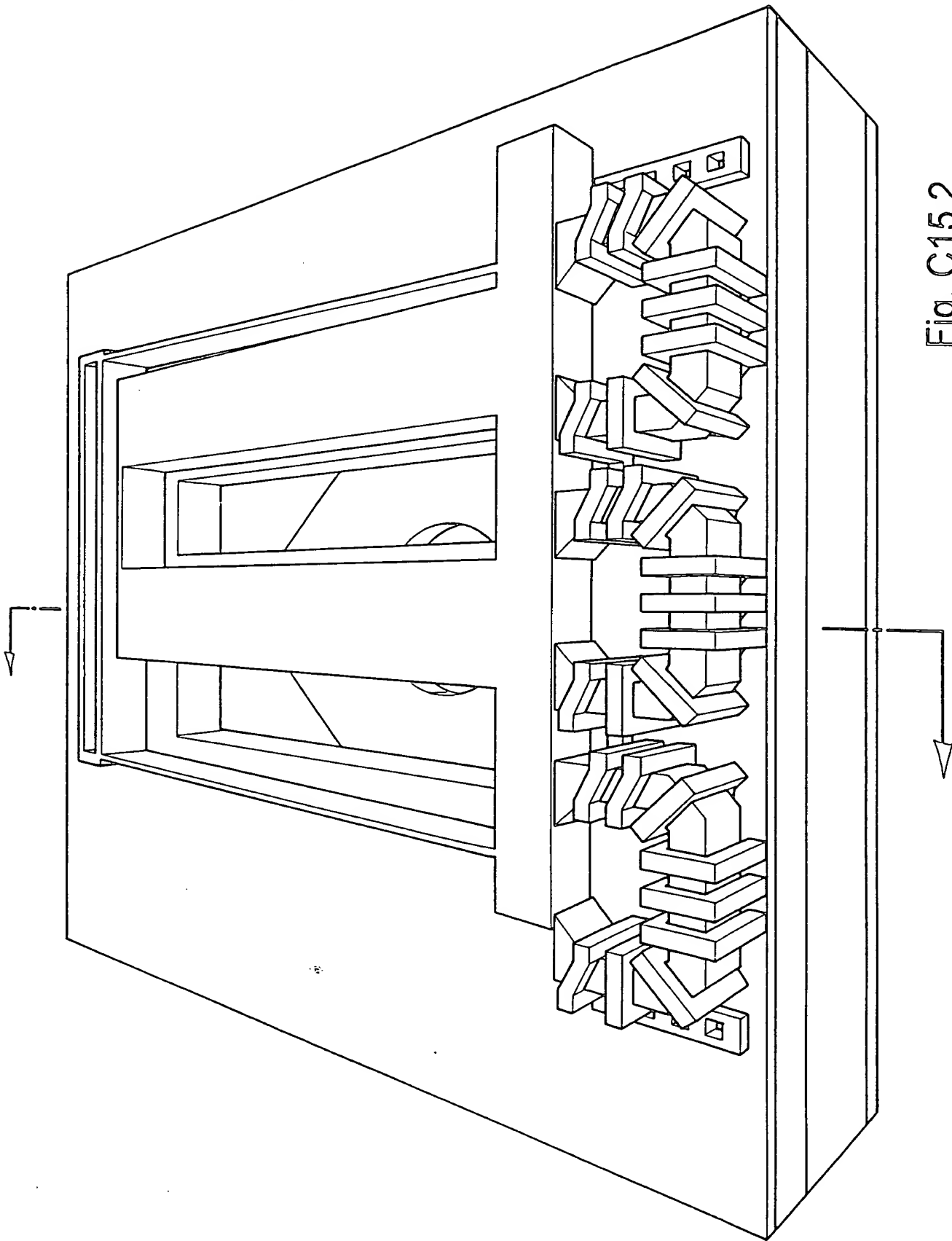


Fig. C15.2

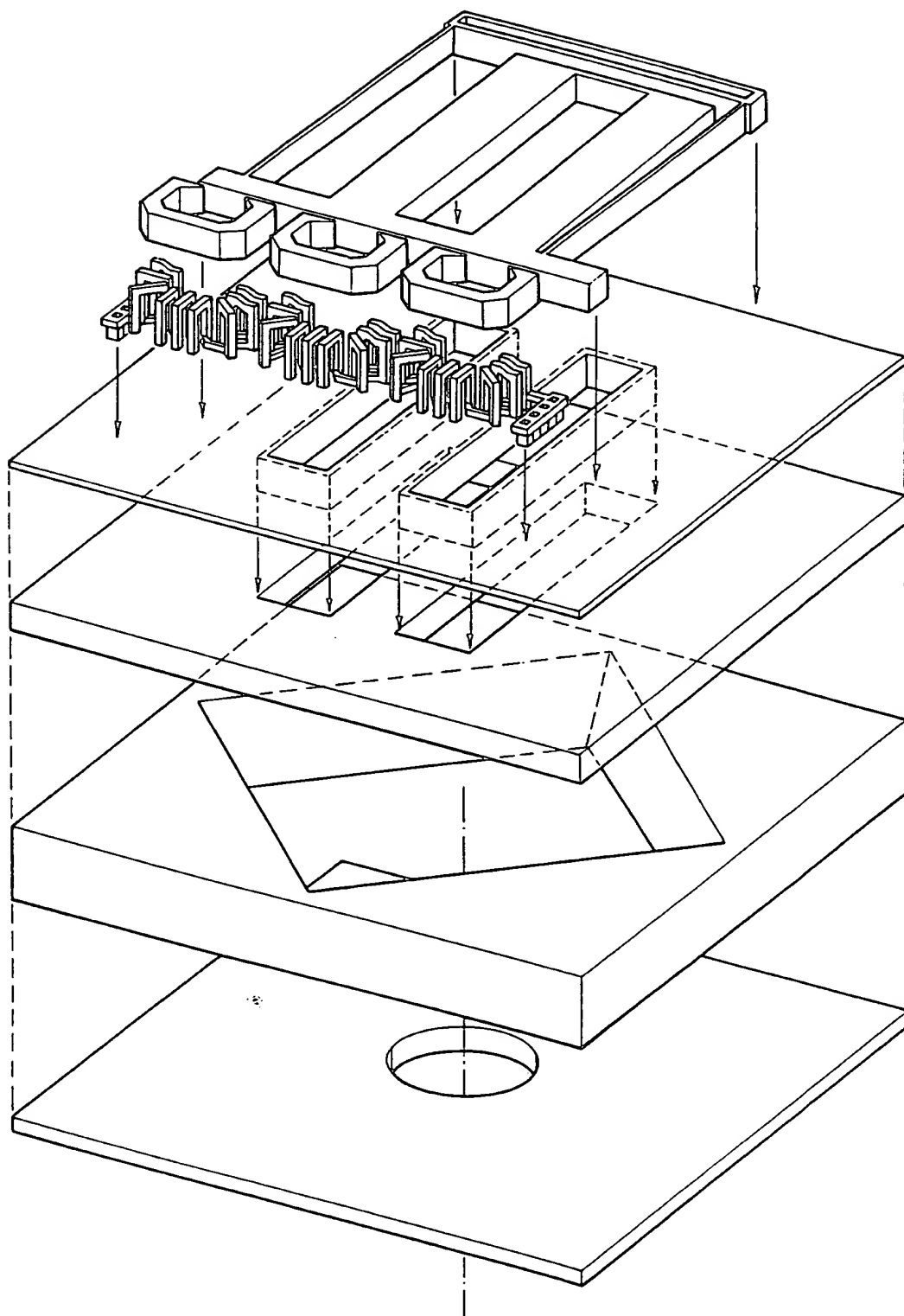


Fig. C15.3

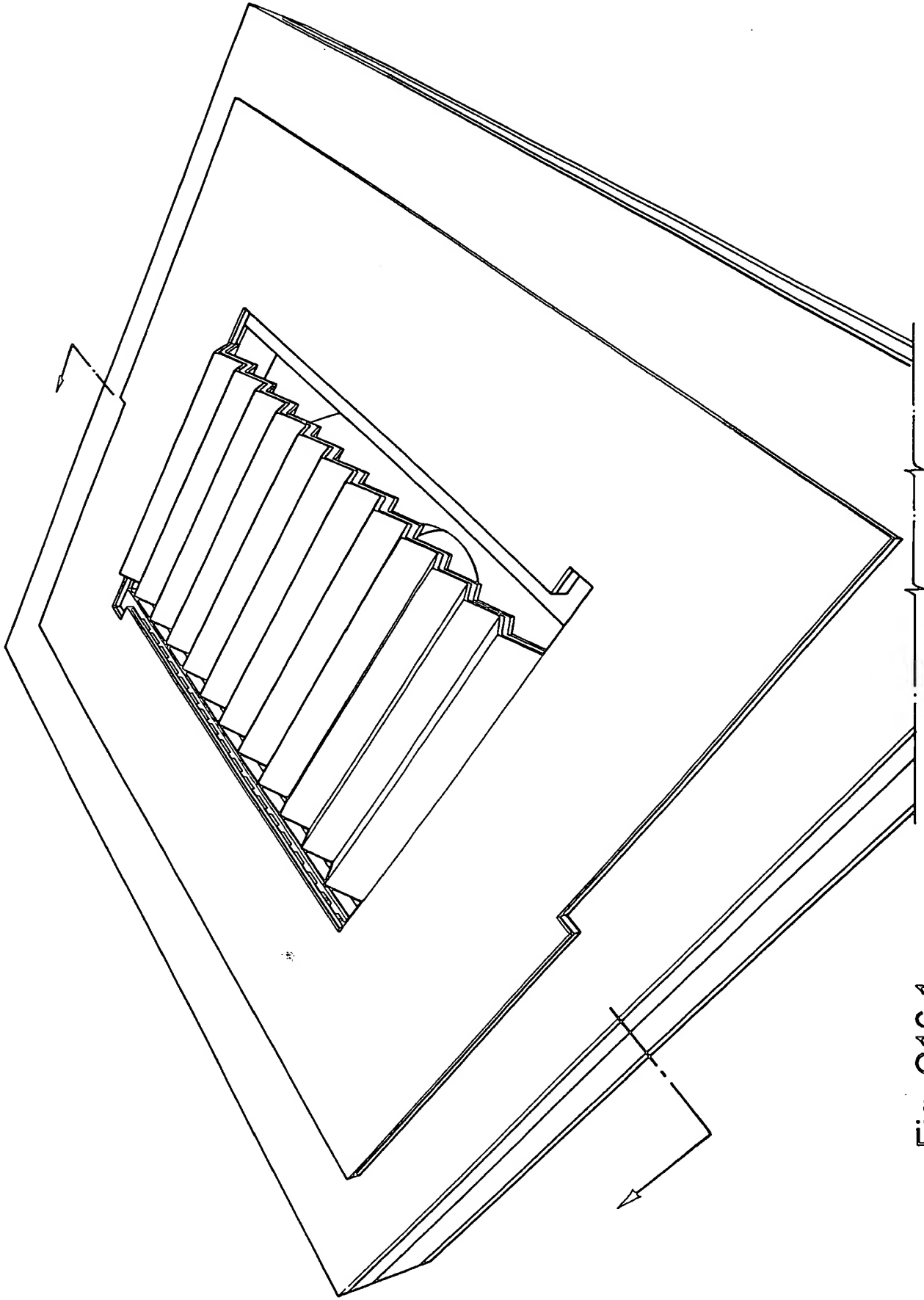


Fig. C16.1

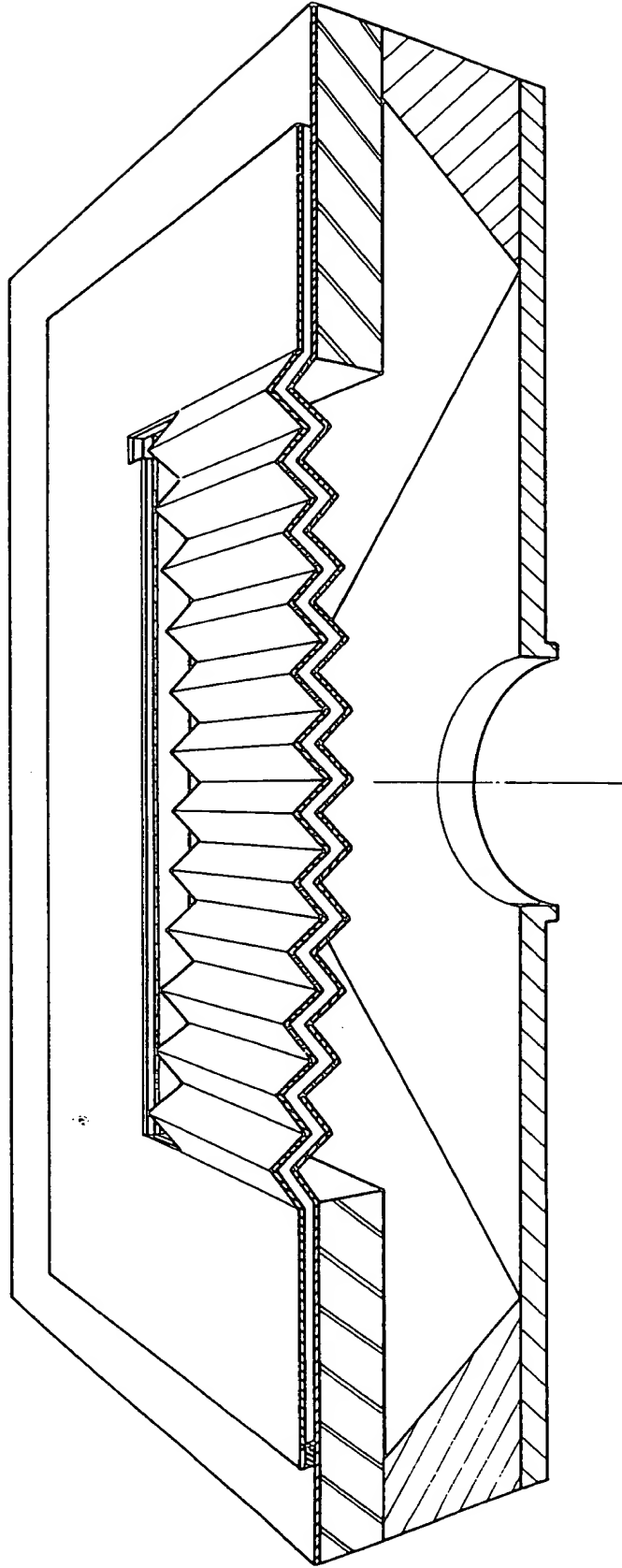


Fig. C16.2

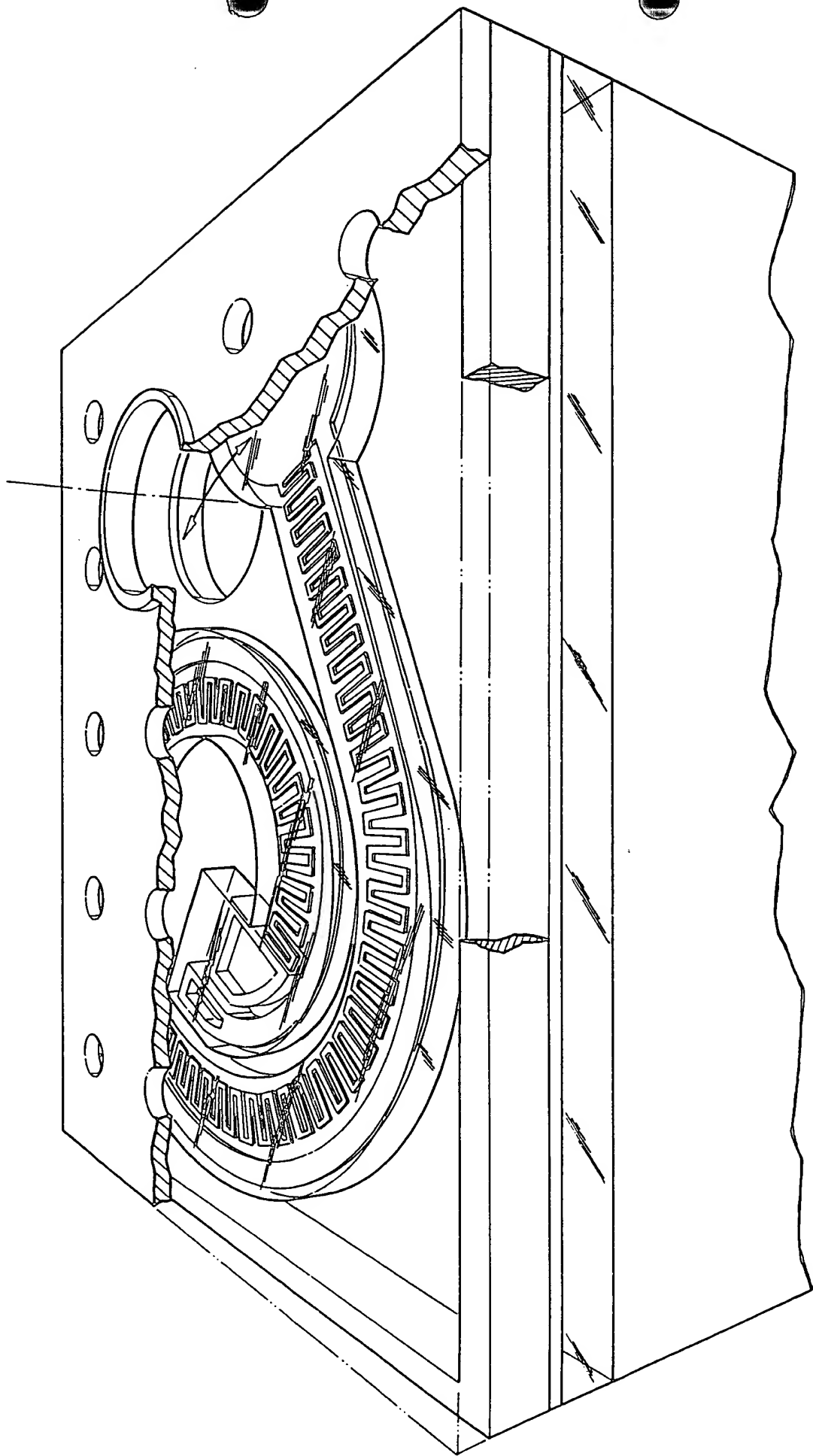


Fig. C17.1

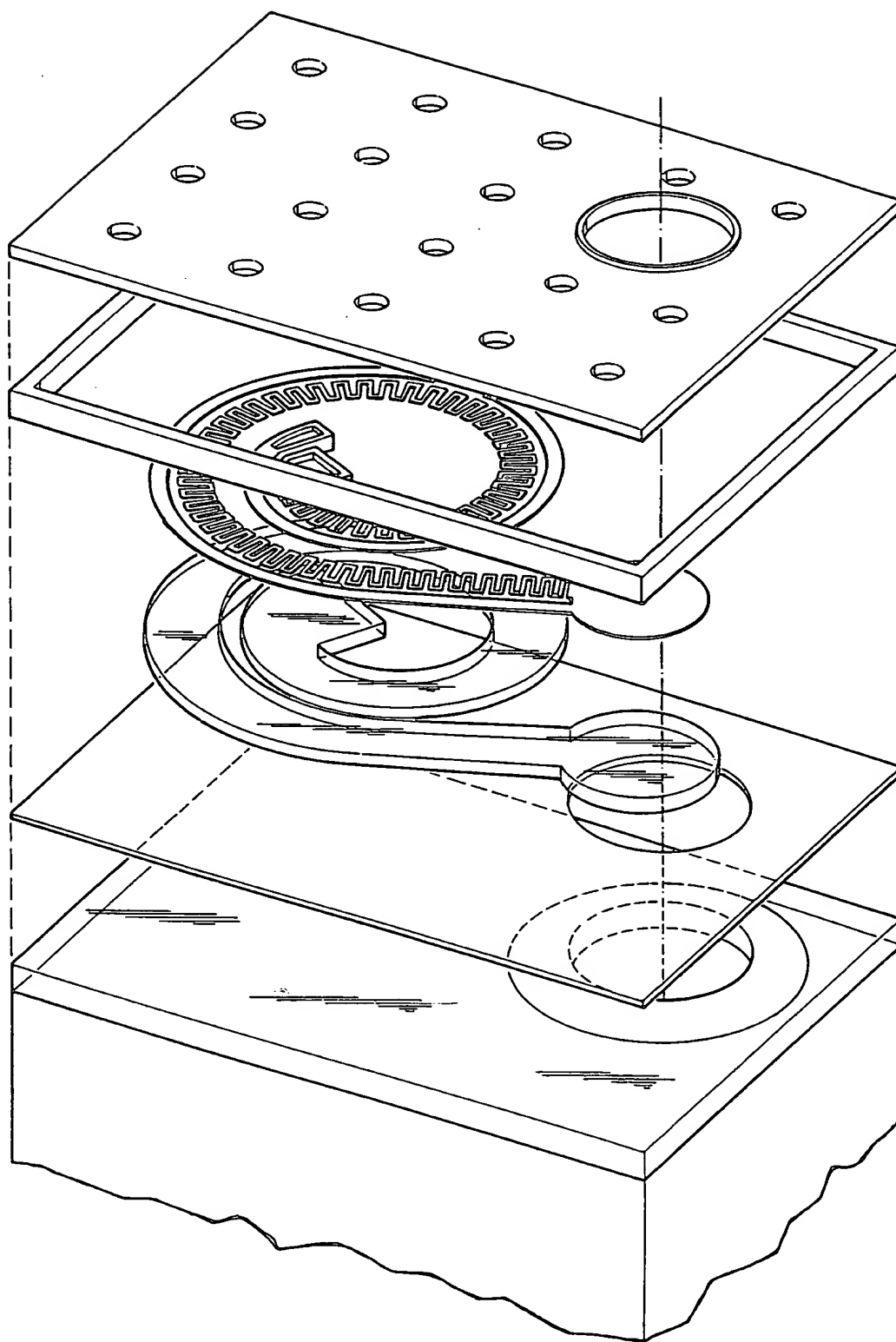


Fig. C17.2

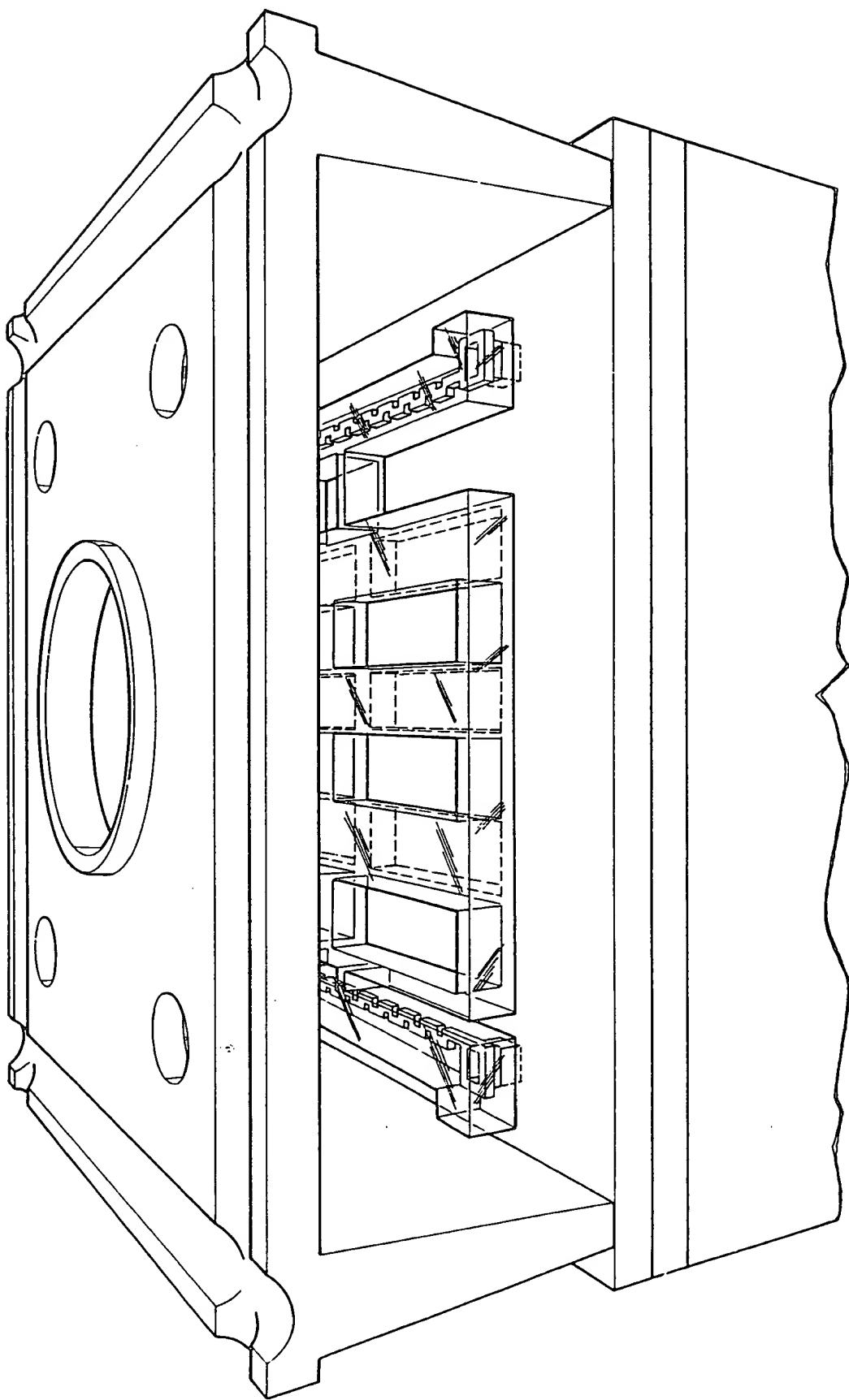


Fig. C18.1

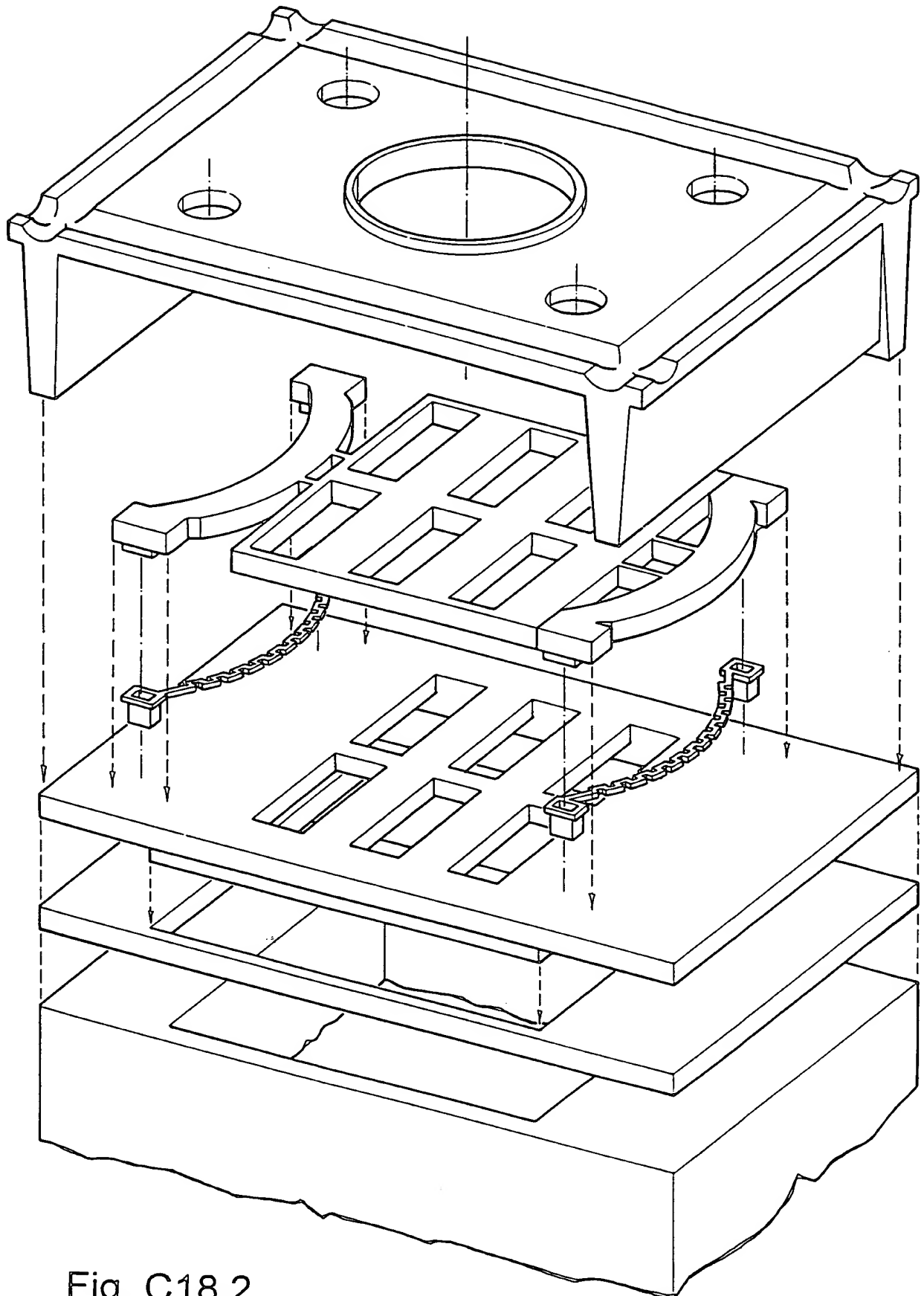


Fig. C18.2

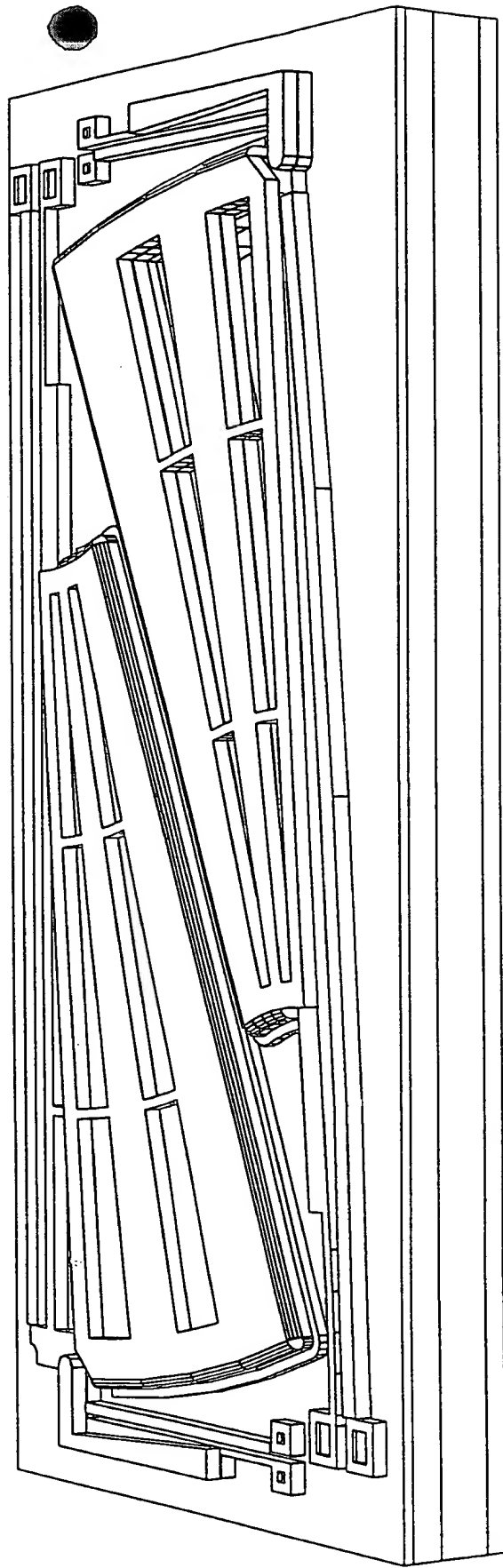


Fig. C19.1

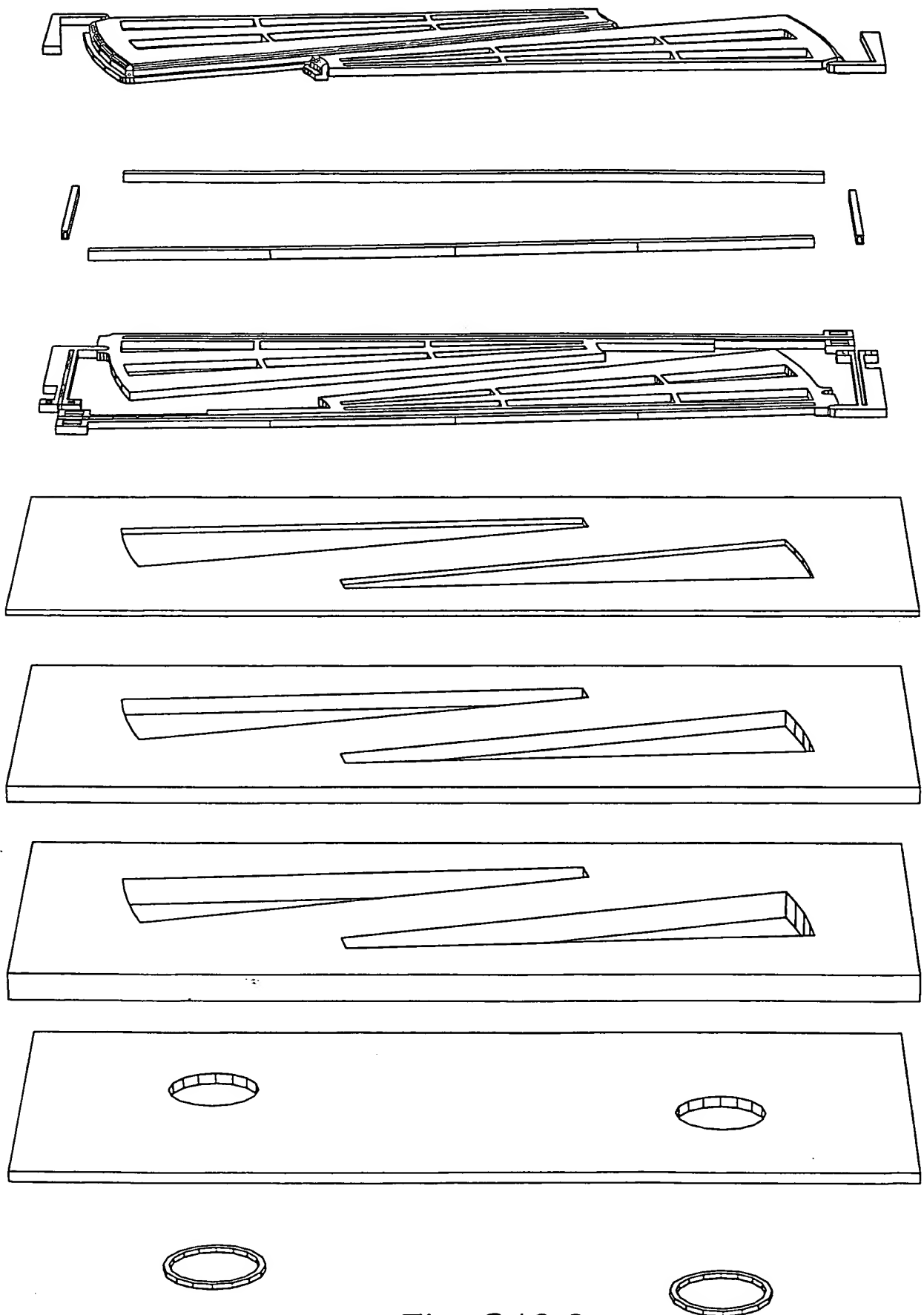


Fig. C19.2

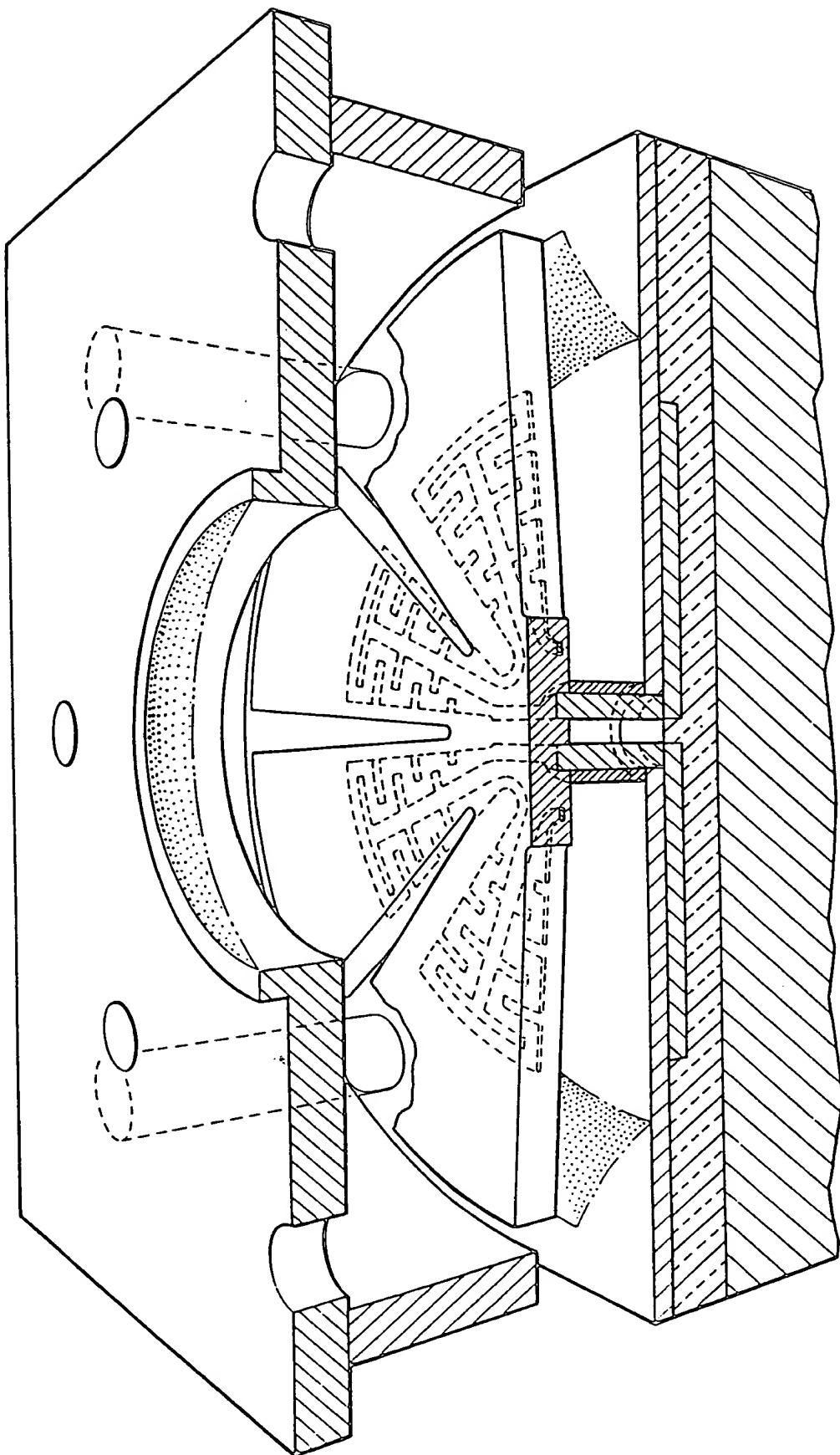


Fig. C20.1

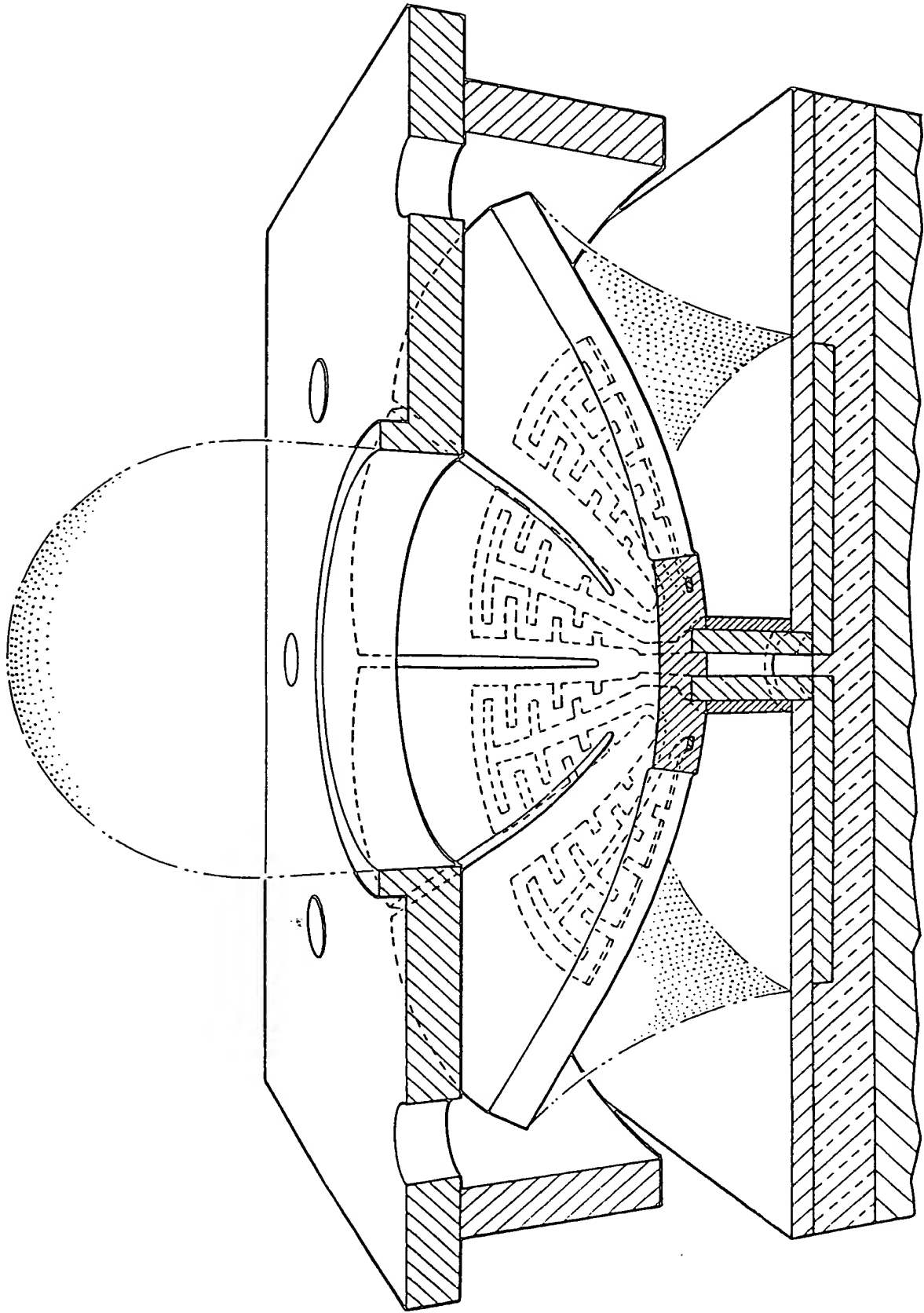


Fig. C20.2

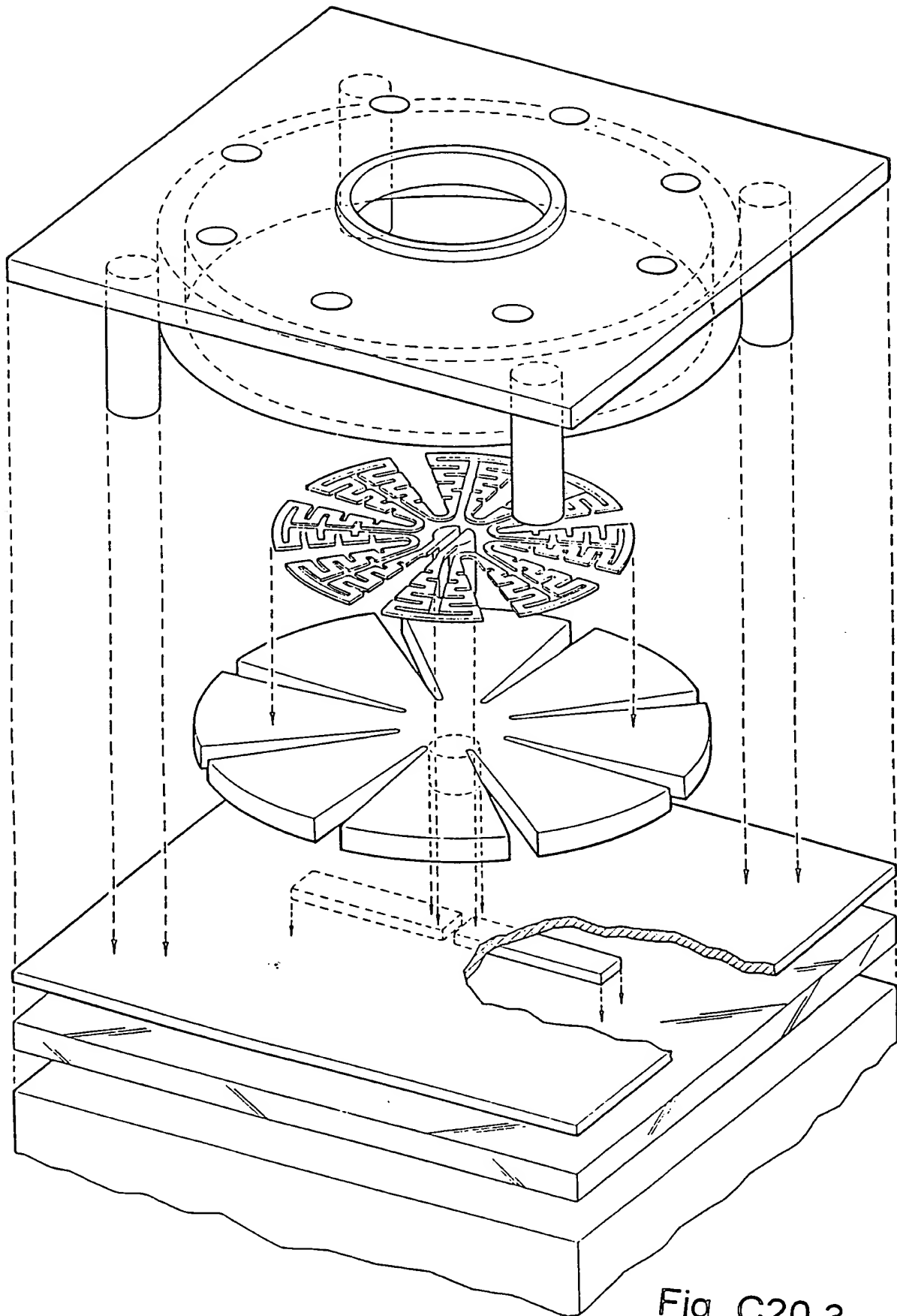


Fig. C20.3

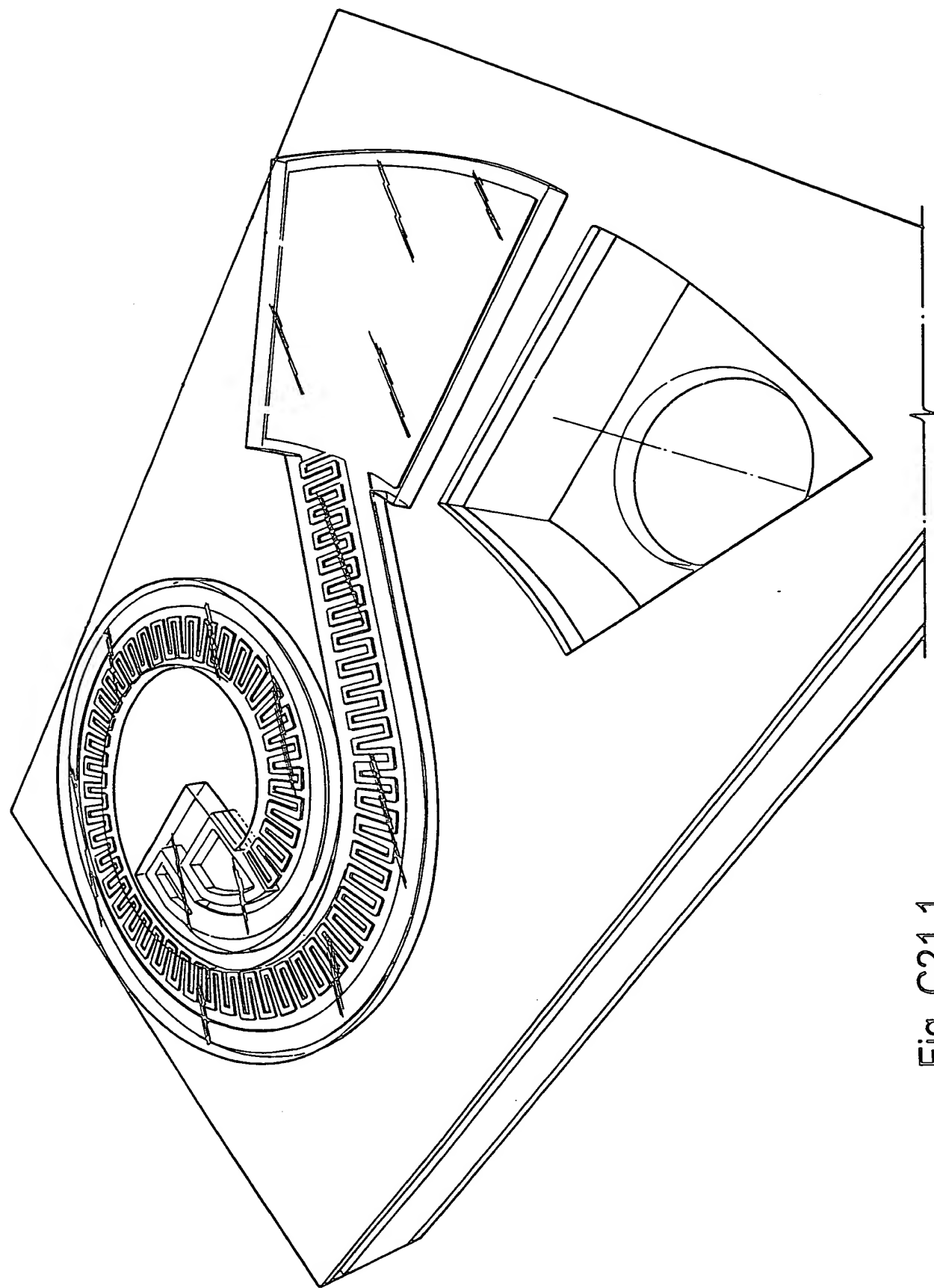


Fig. C21.1

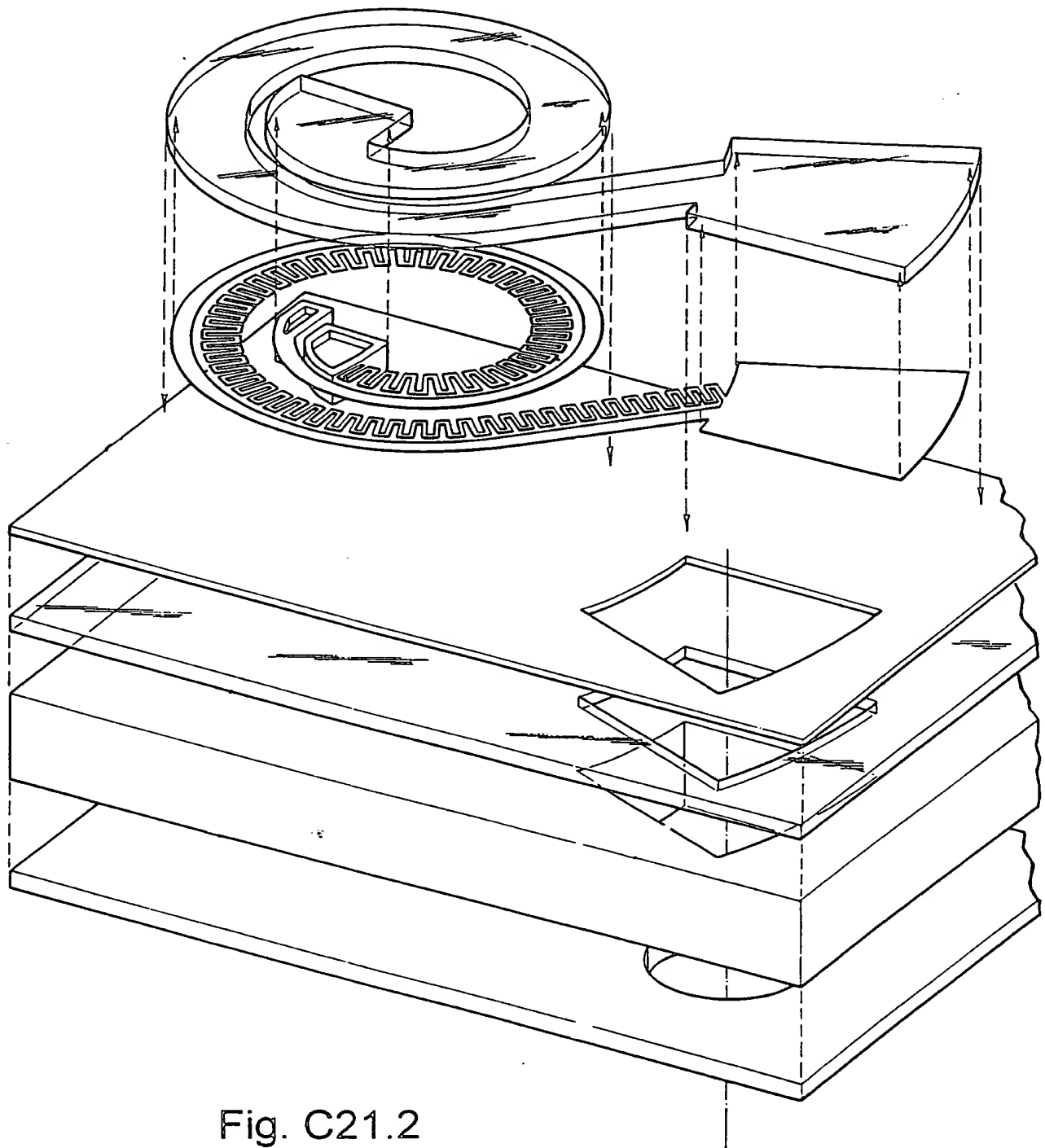


Fig. C21.2

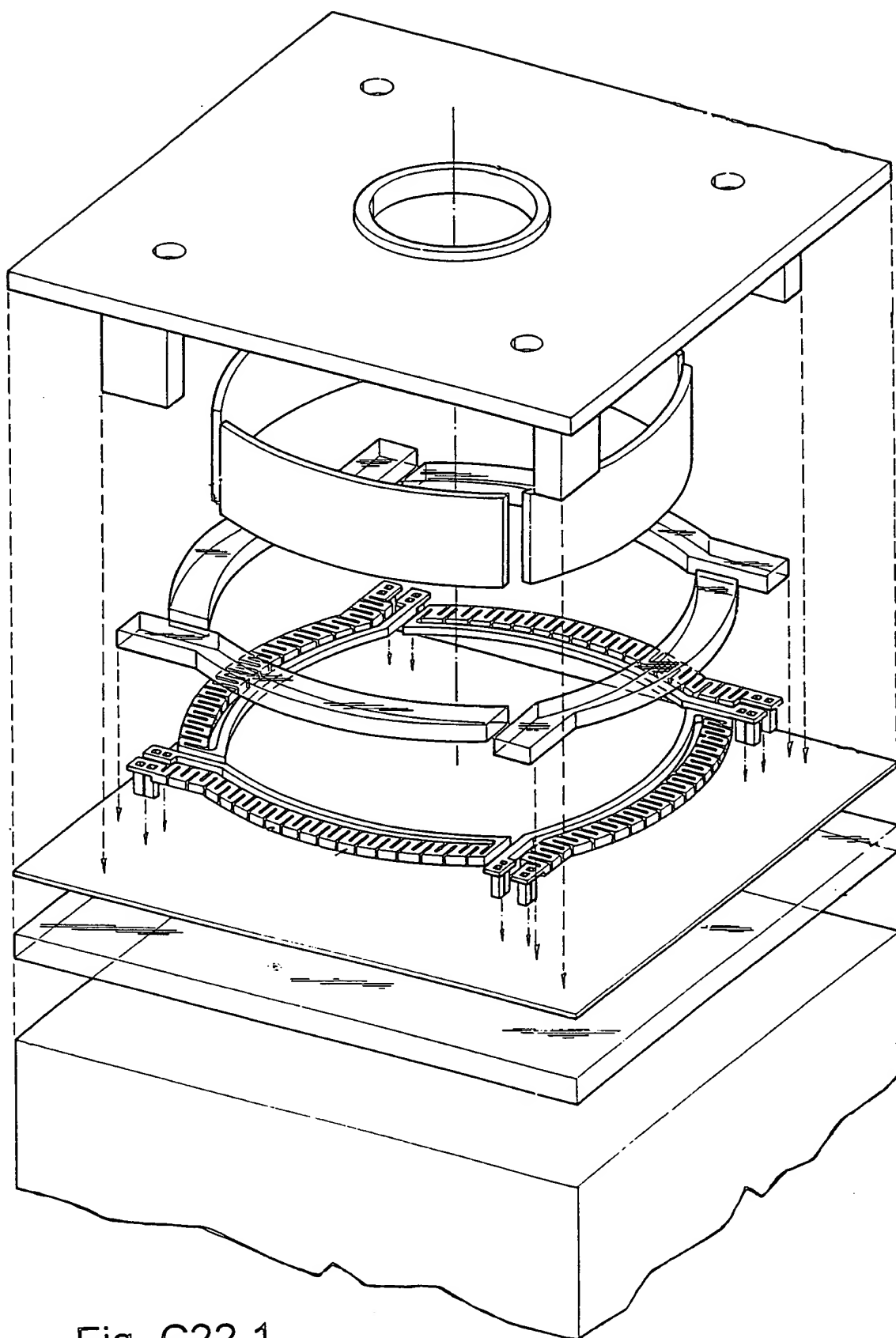


Fig. C22.1

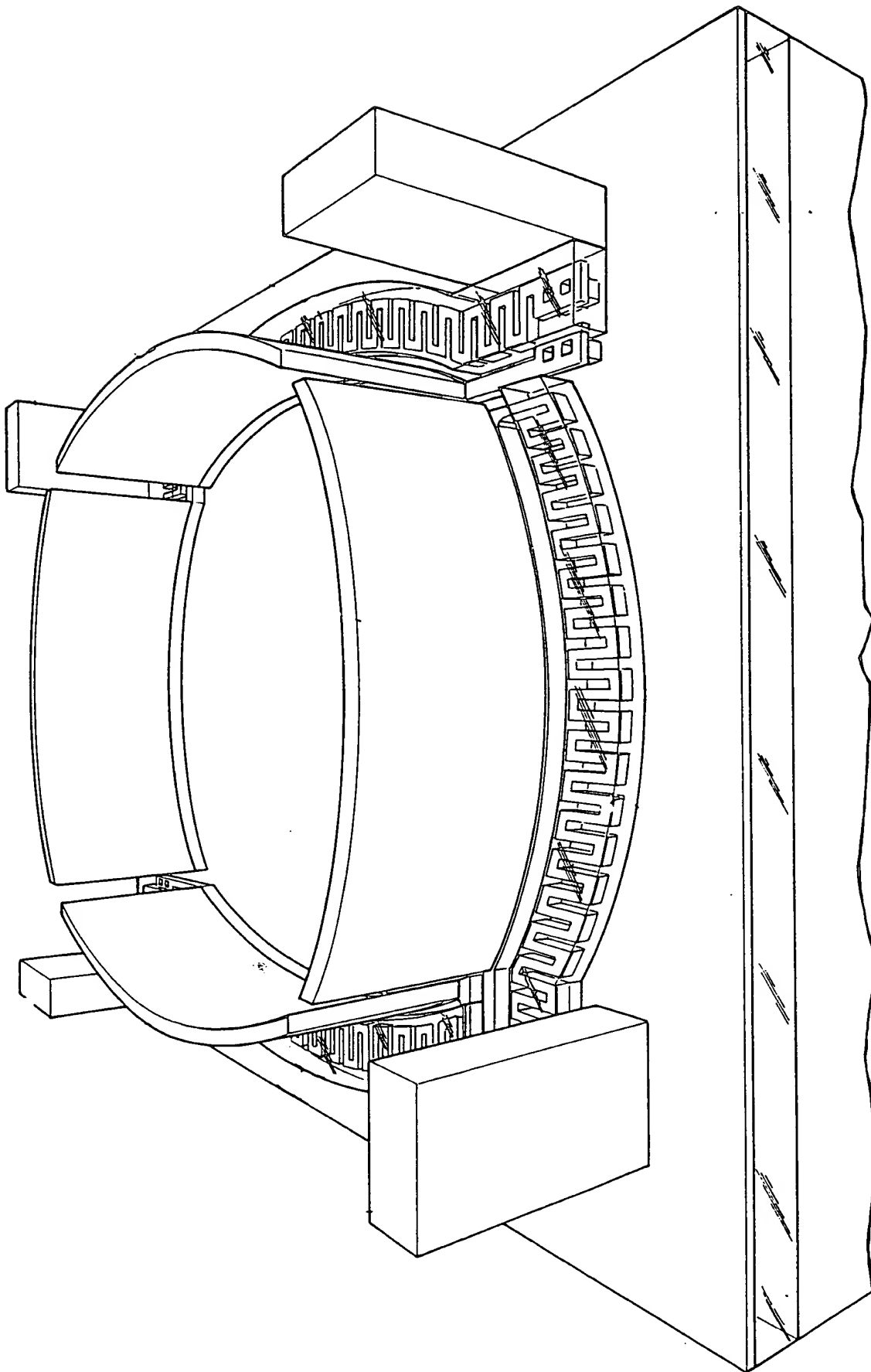


Fig. C22.2

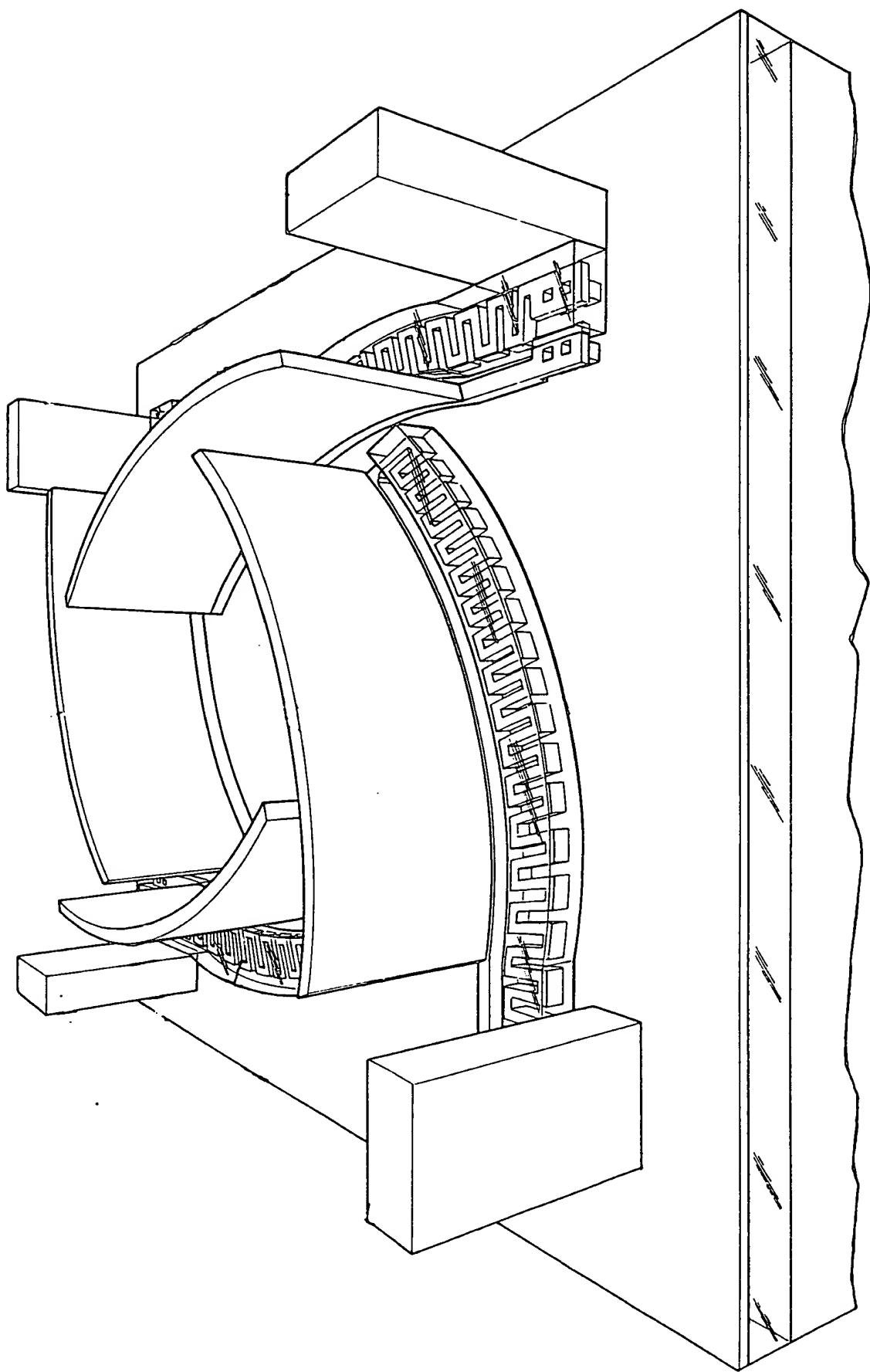


Fig. C22.3

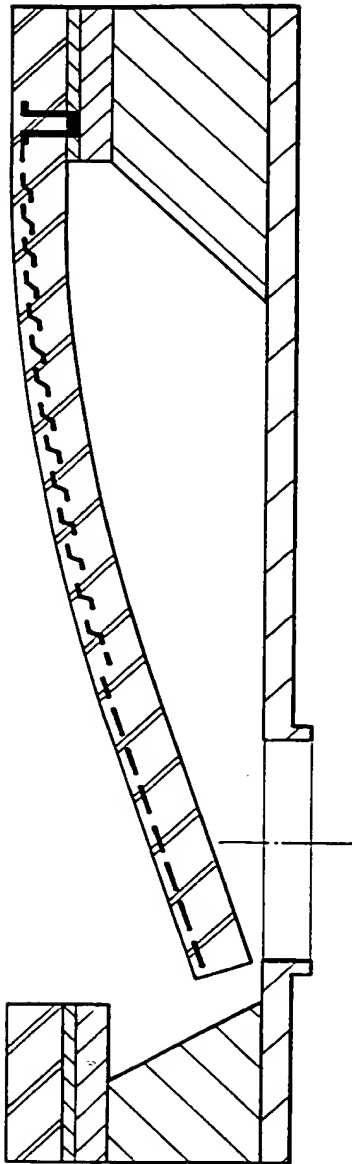


Fig. C23.1

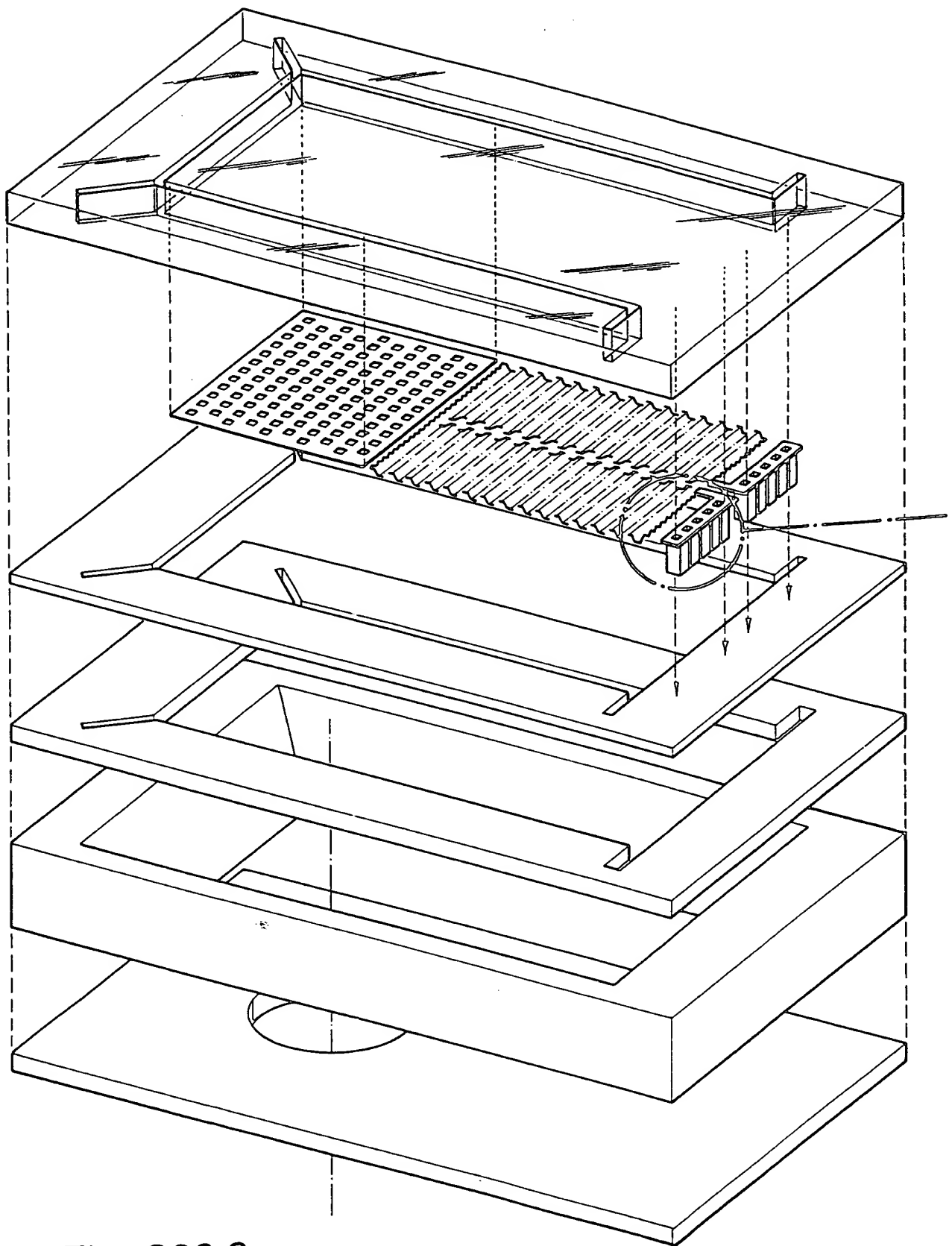


Fig. C23.2

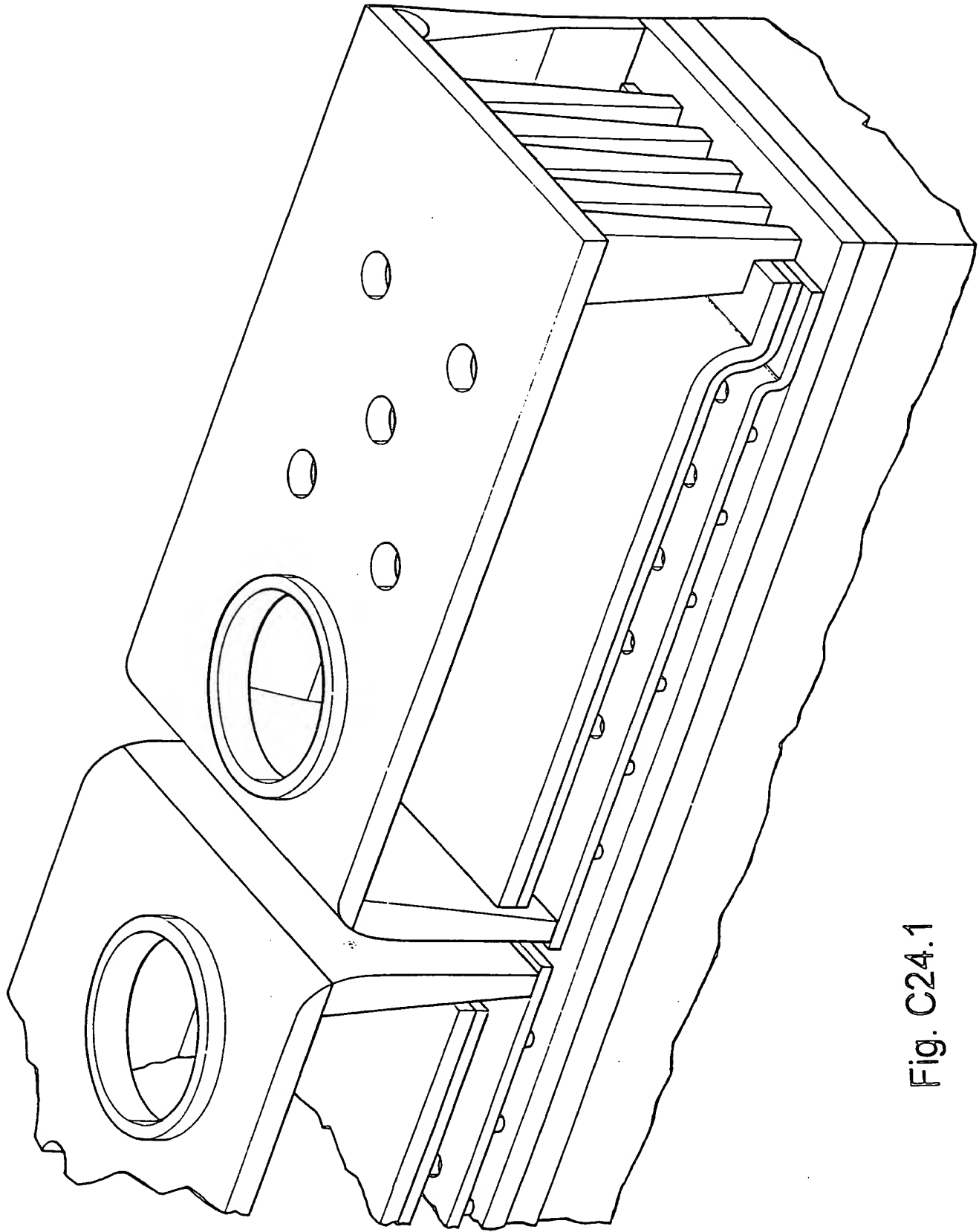


Fig. C24.1

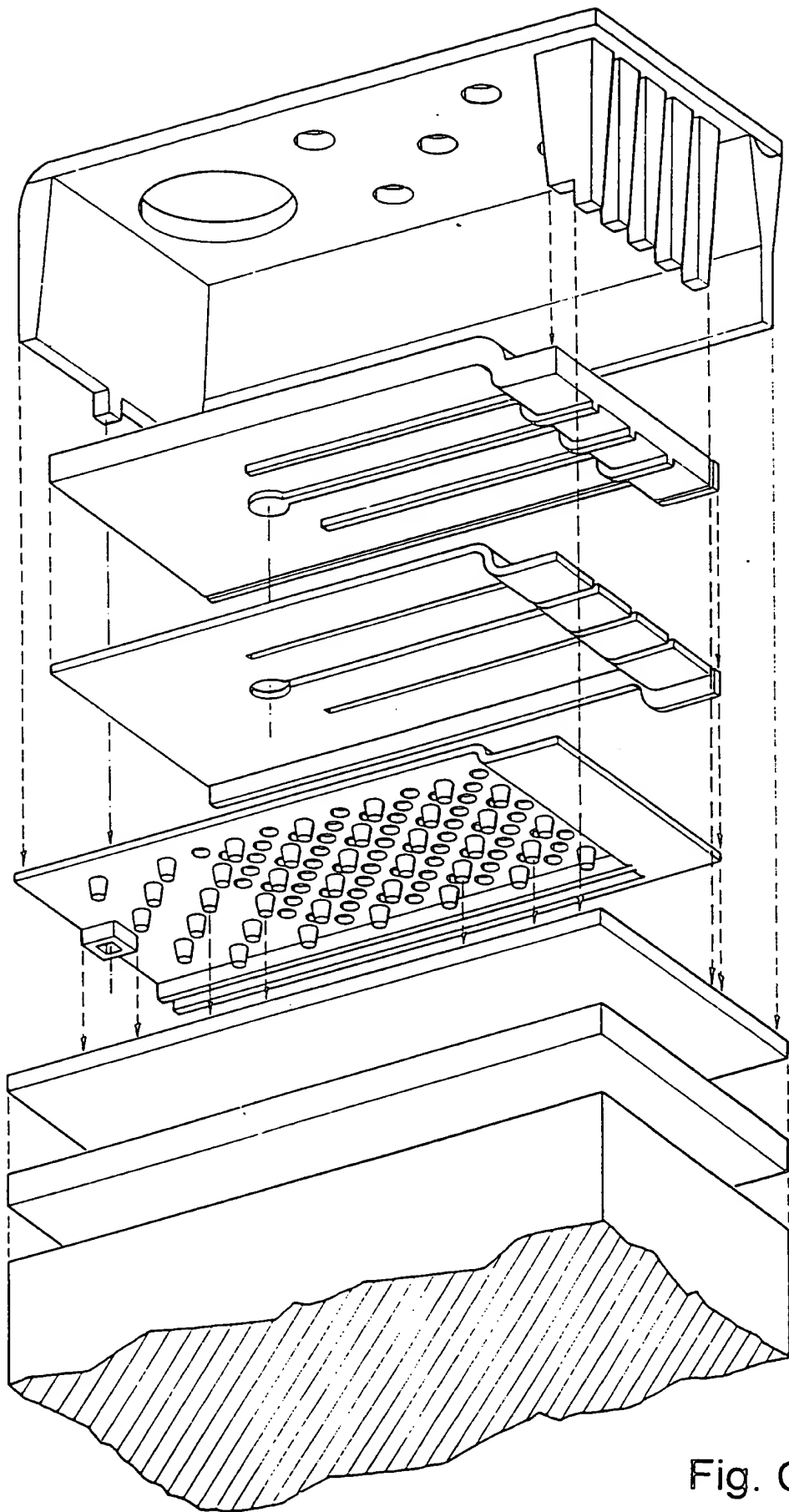


Fig. C24.2

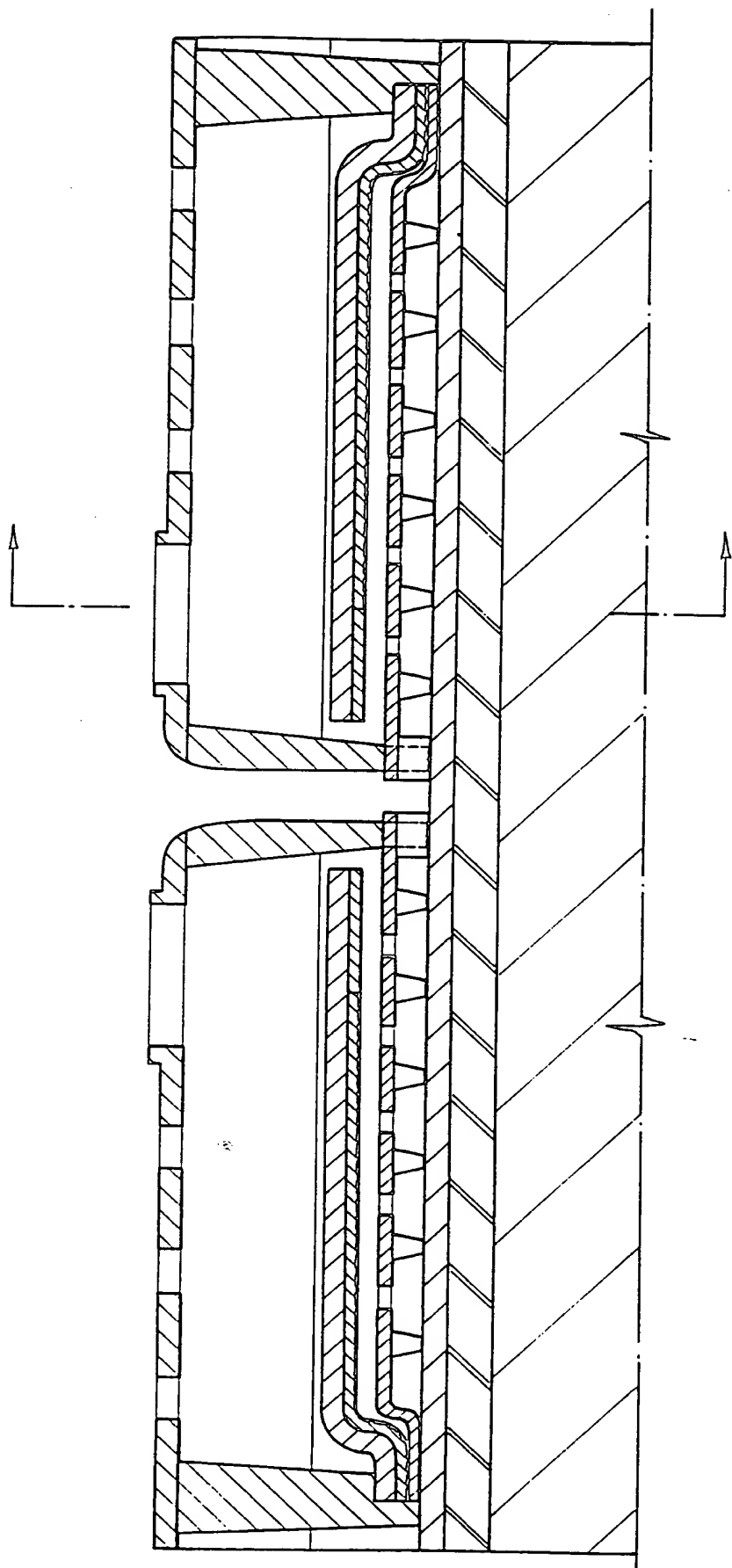


Fig. C24.3

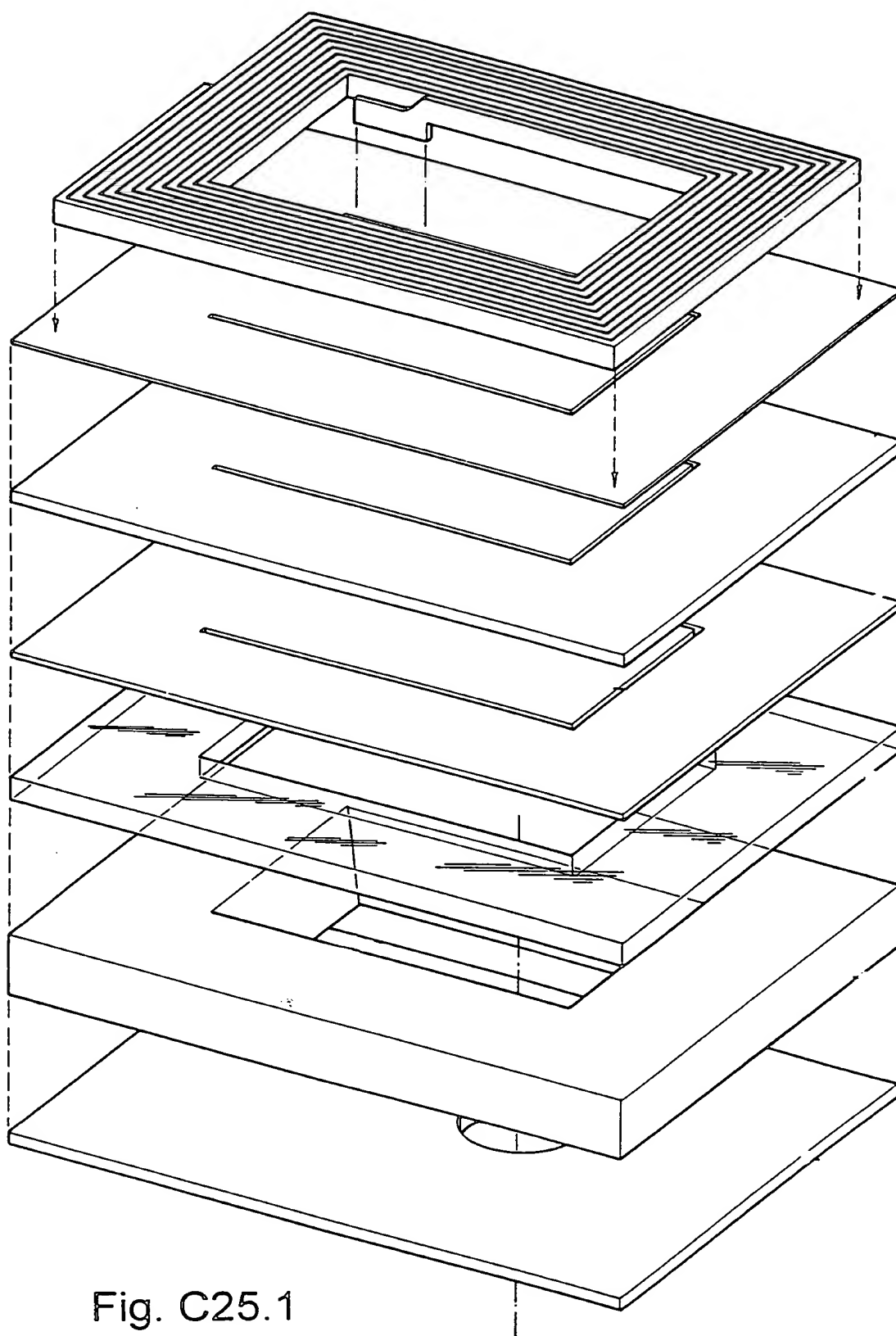


Fig. C25.1

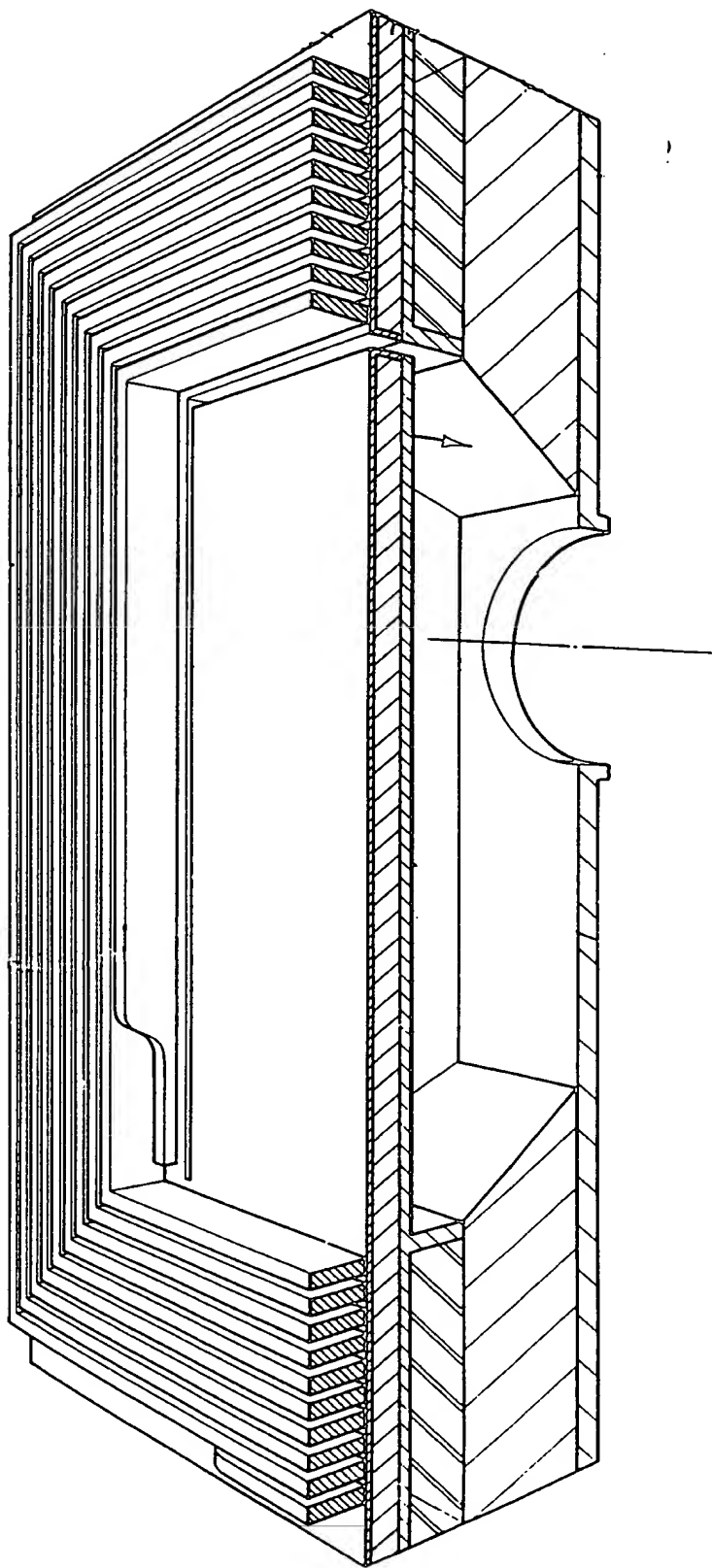


Fig. C25.2

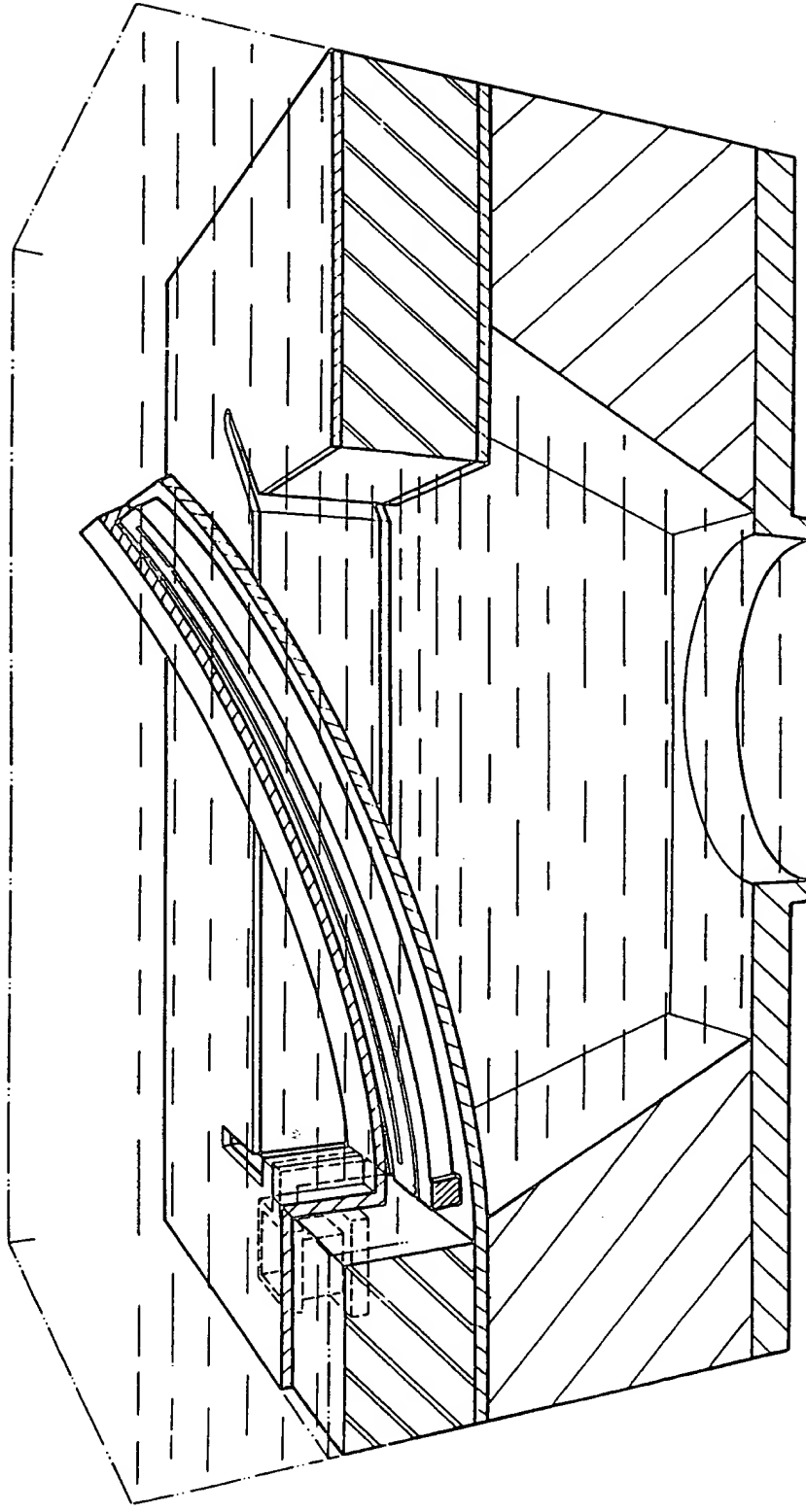


Fig. C26.1

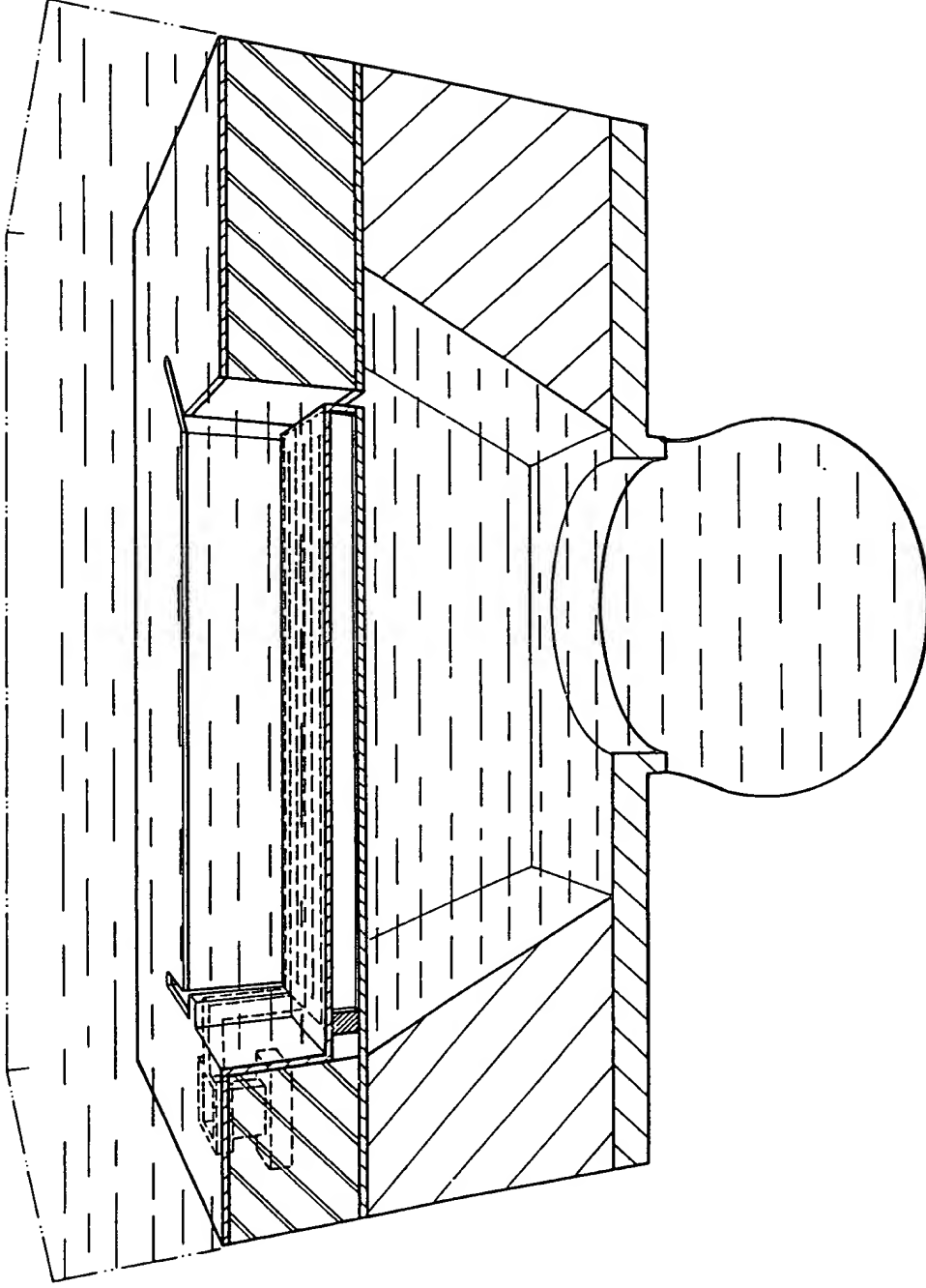


Fig. C26.2

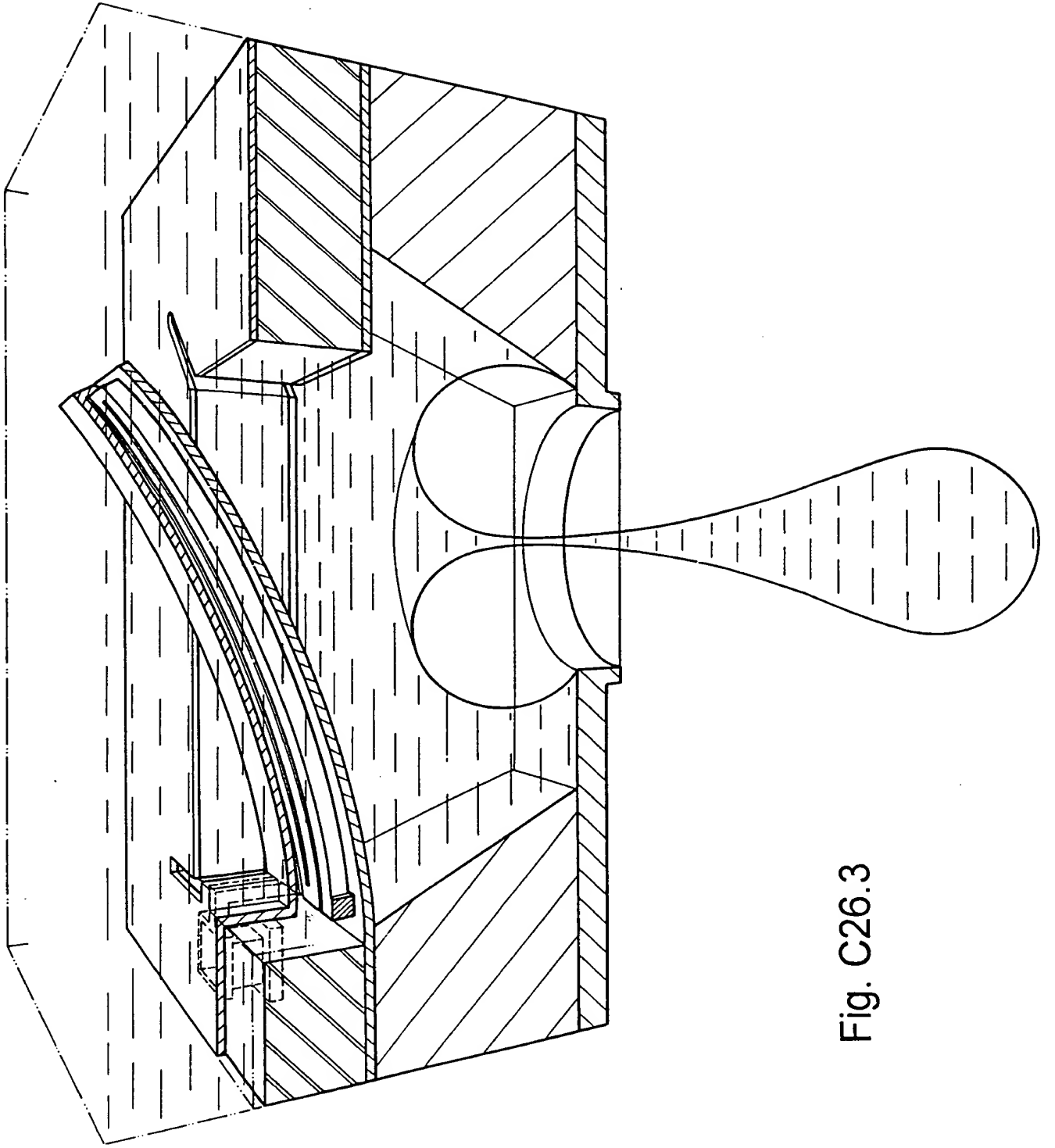


Fig. C26.3

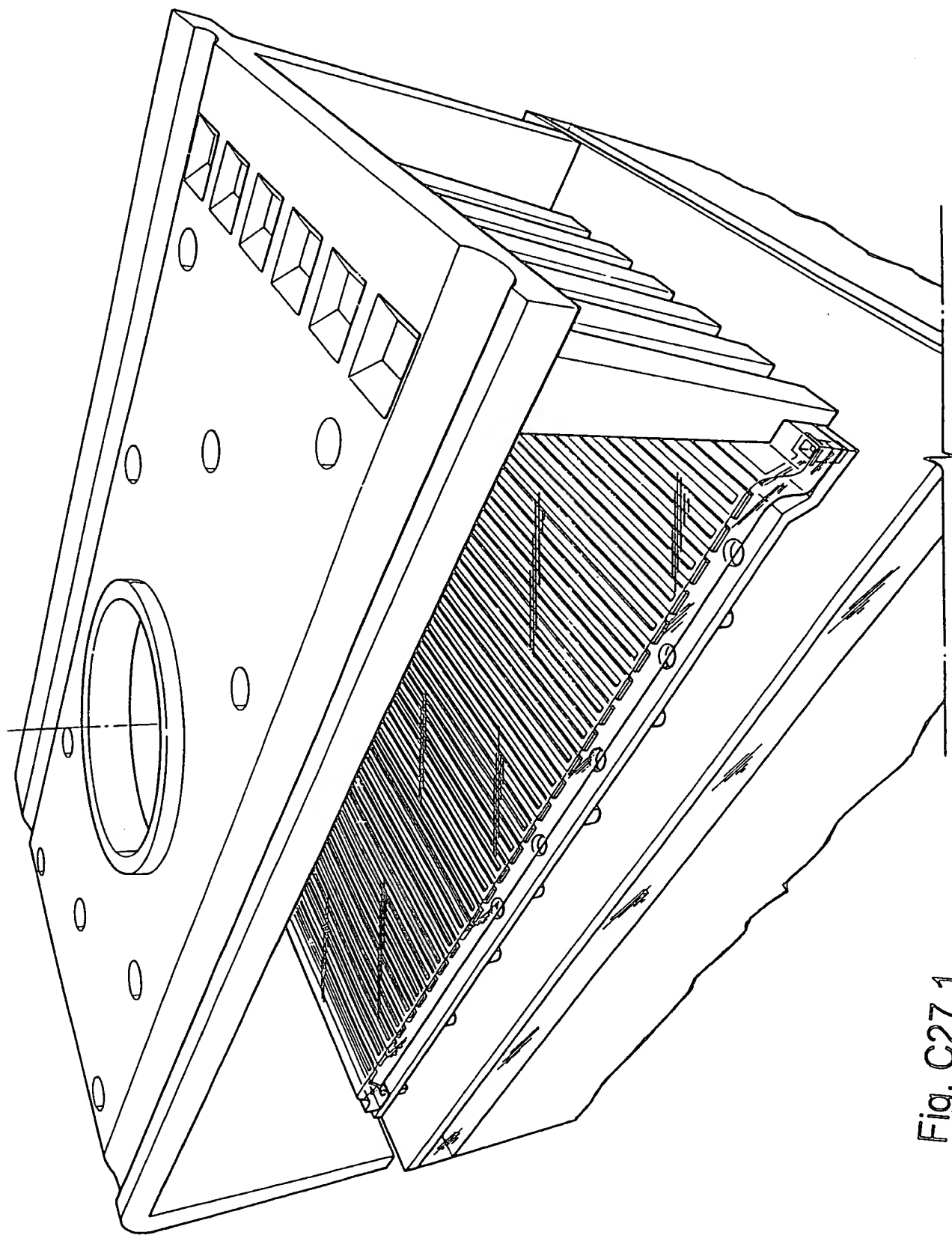


Fig. C27.1

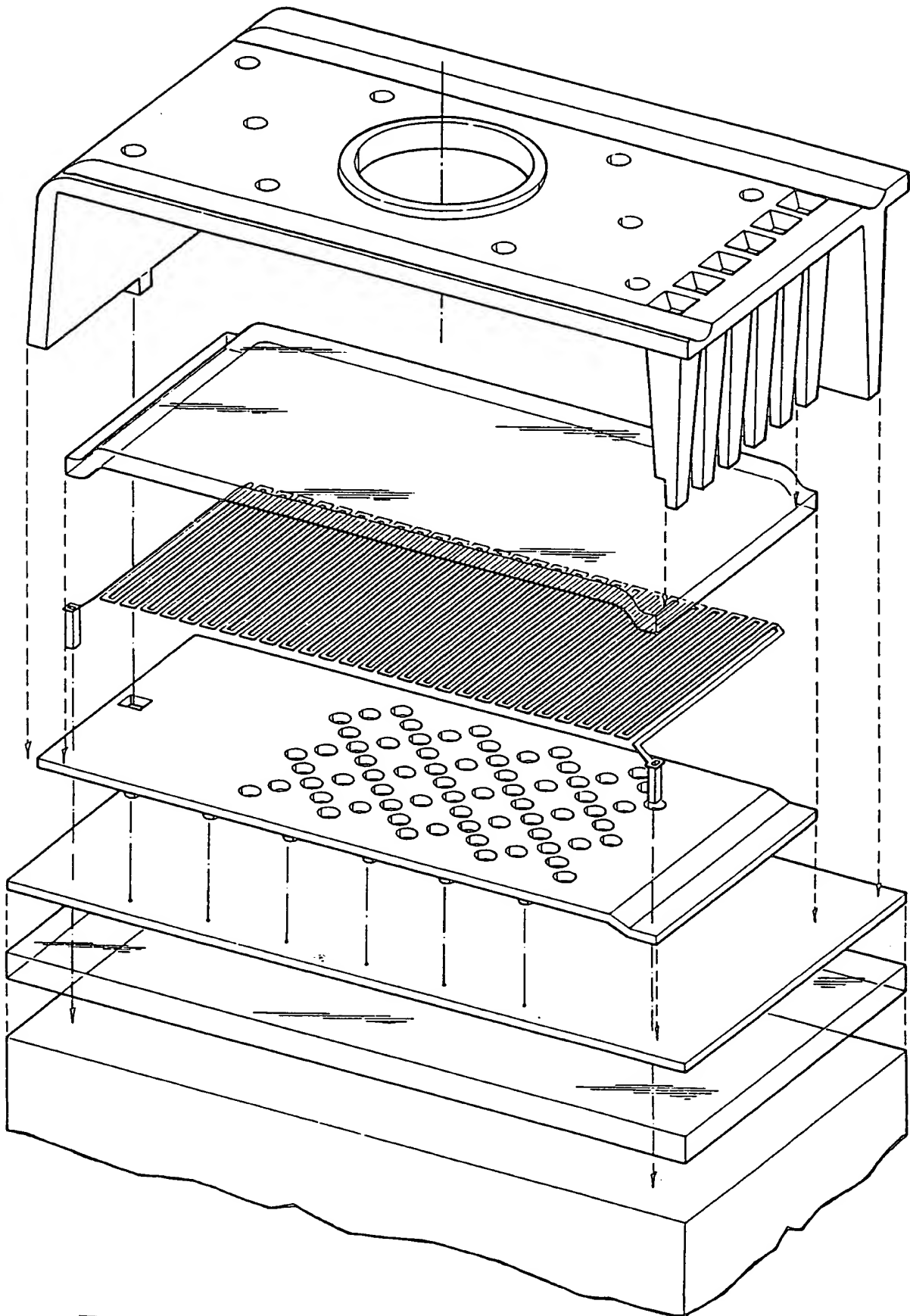


Fig. C27.2

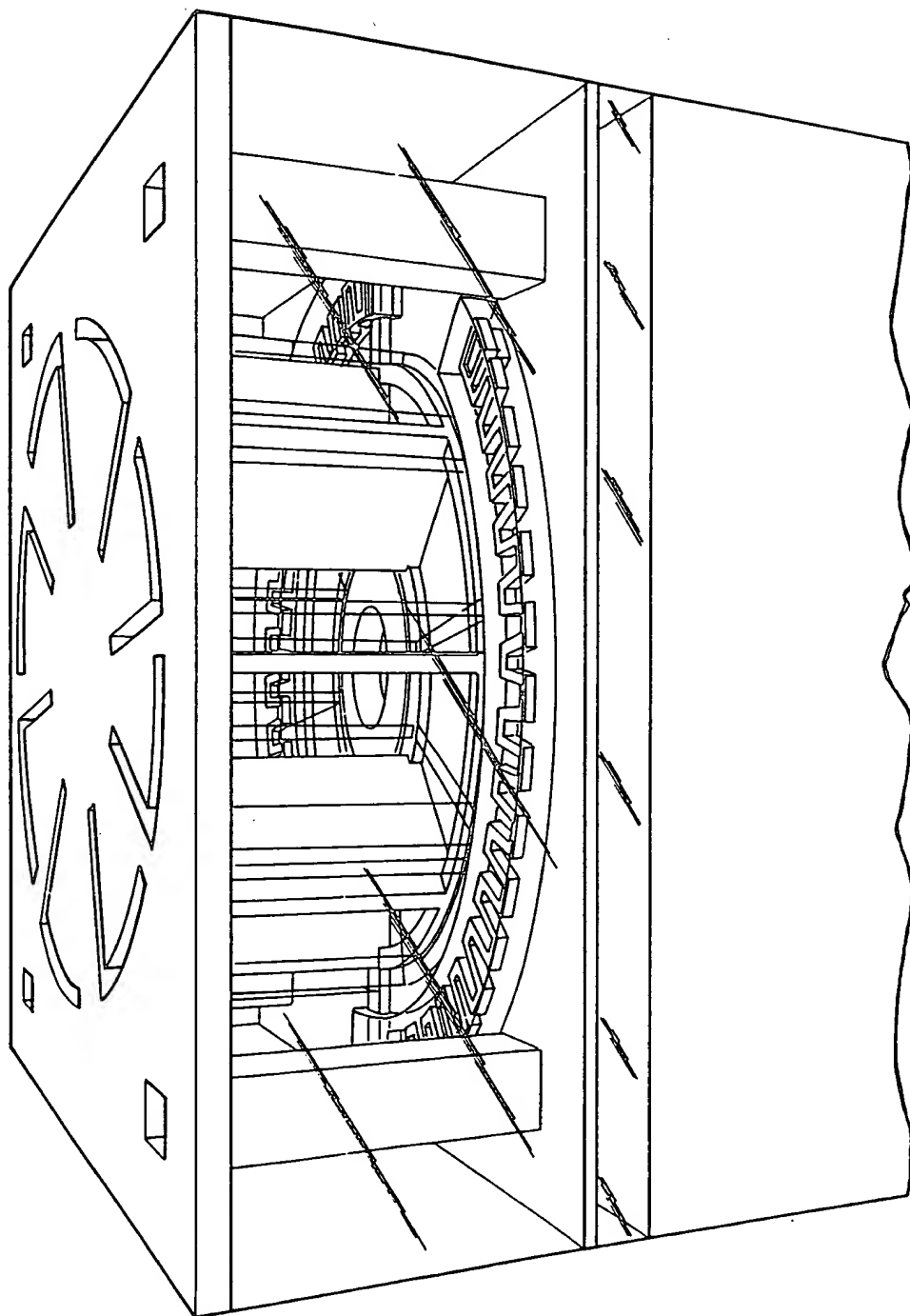


Fig. C28.1

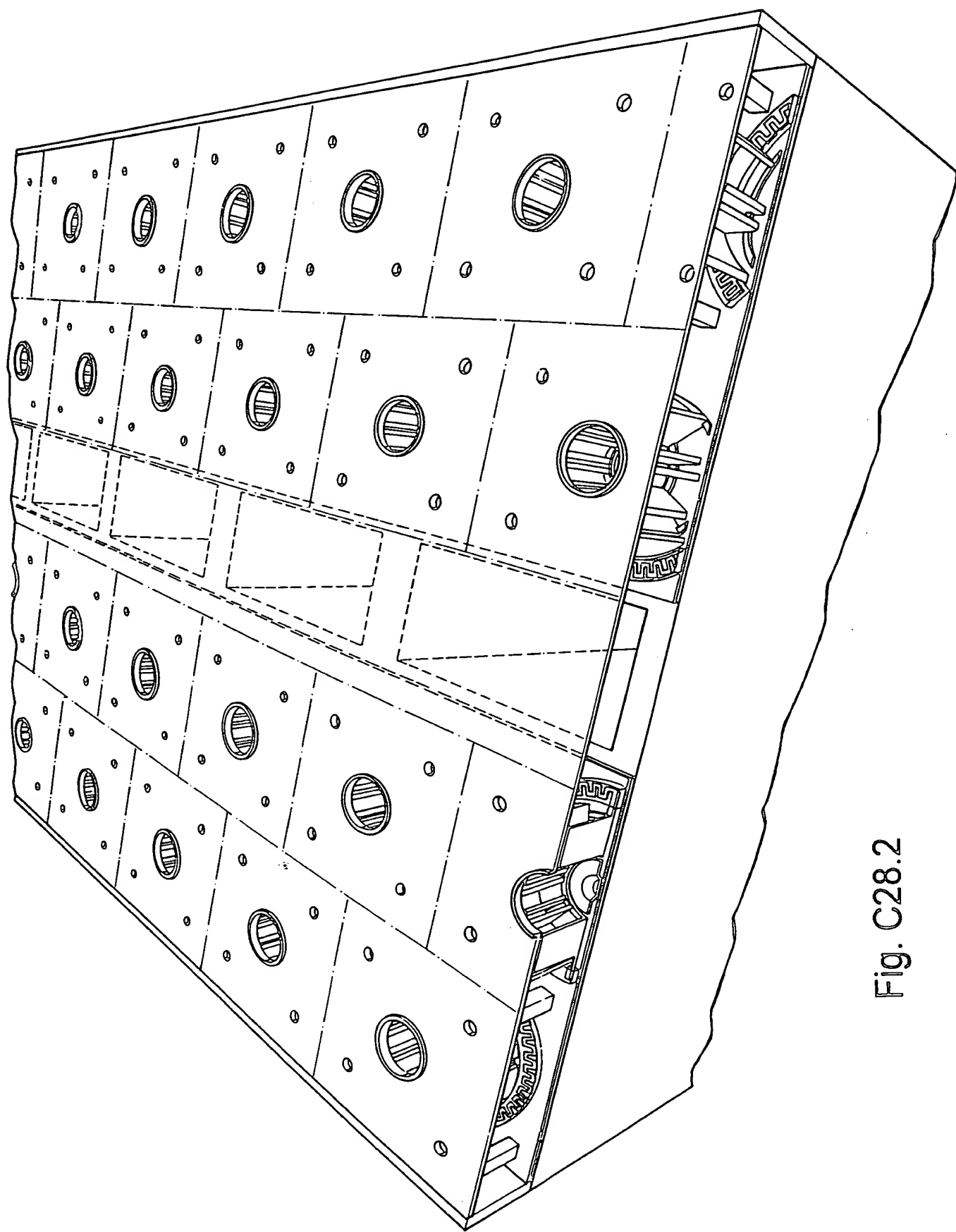


Fig. C28.2

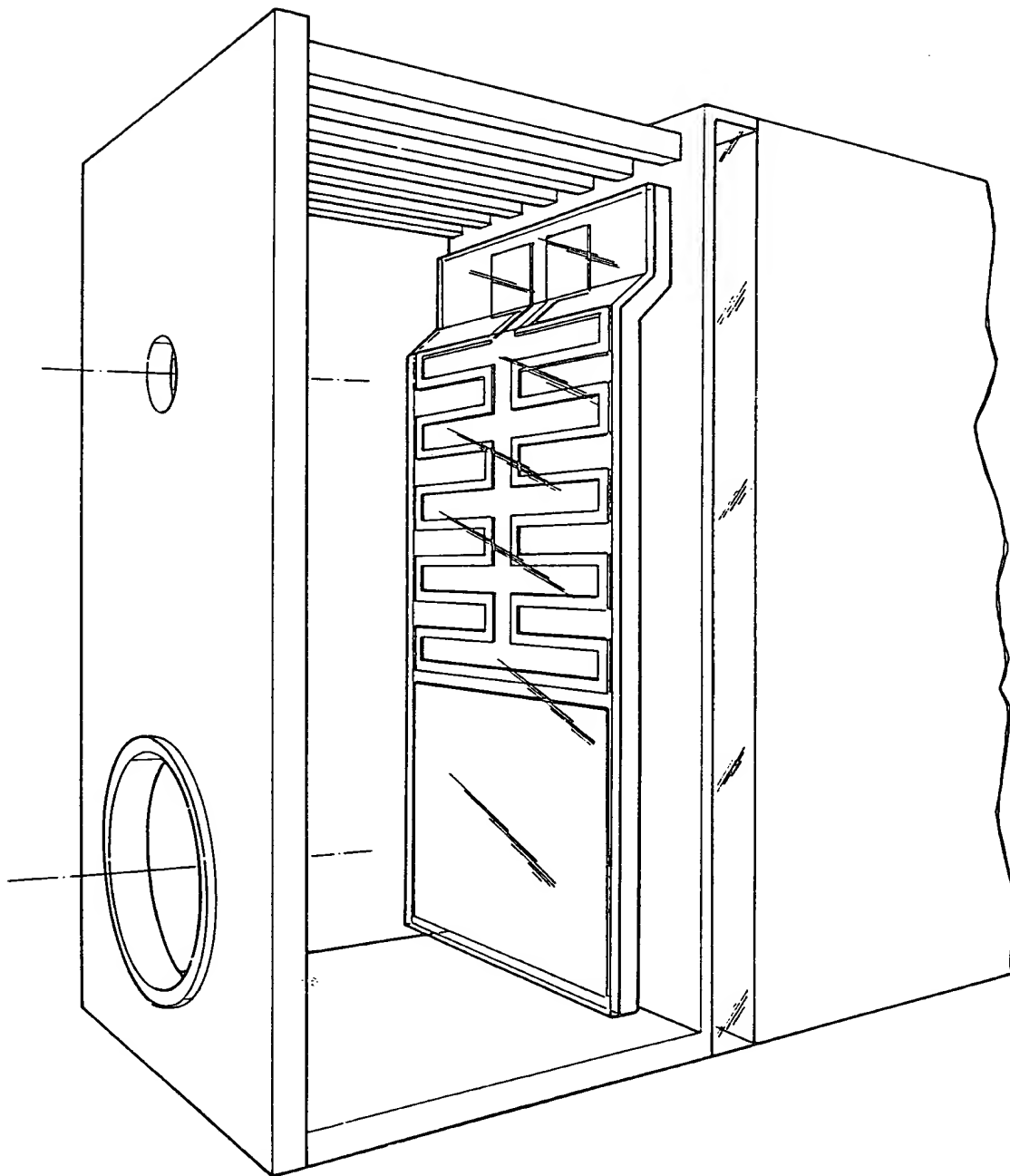


Fig. C29.1

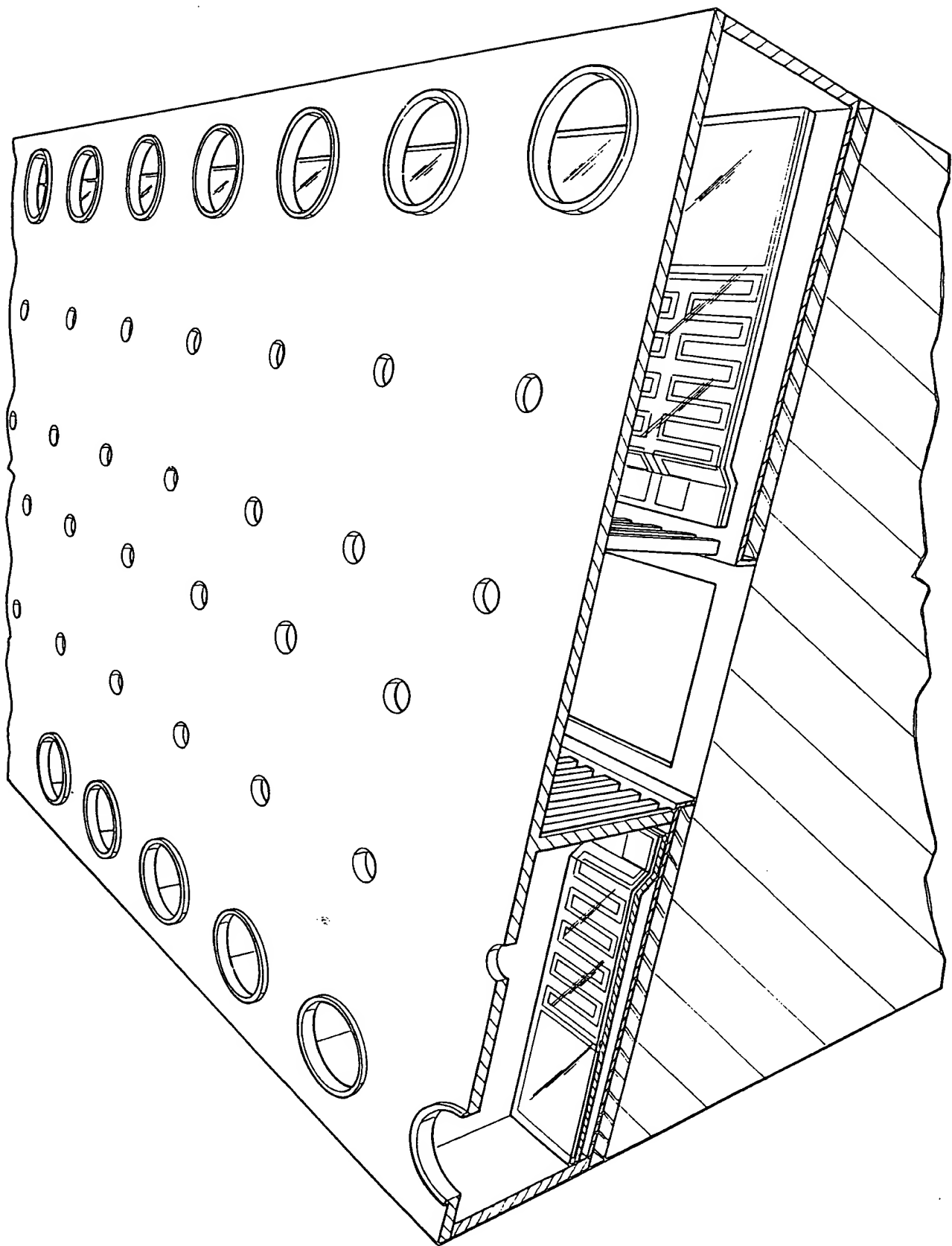


Fig. C29.2

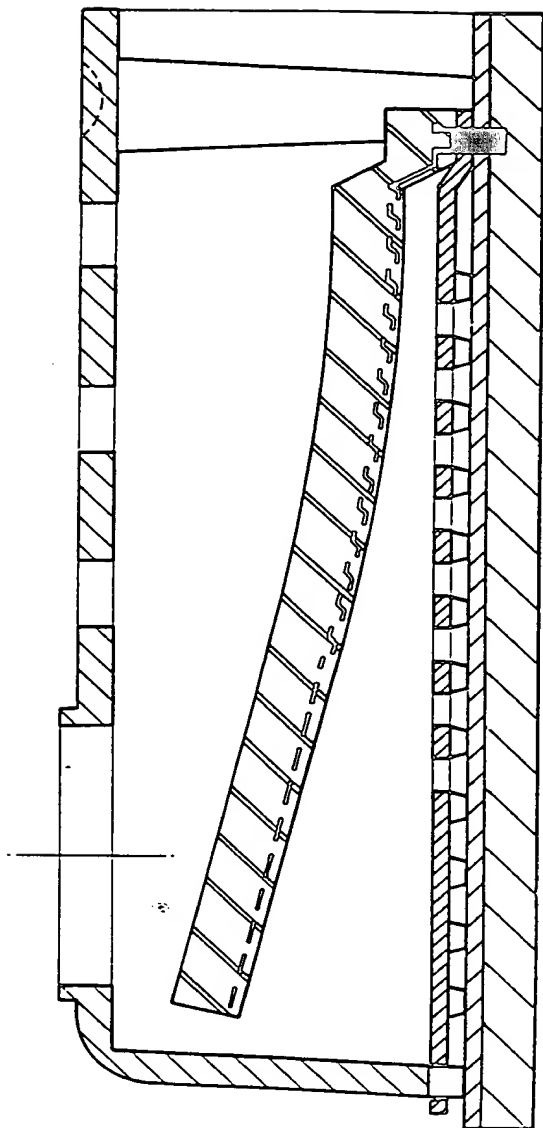


Fig. C30.1

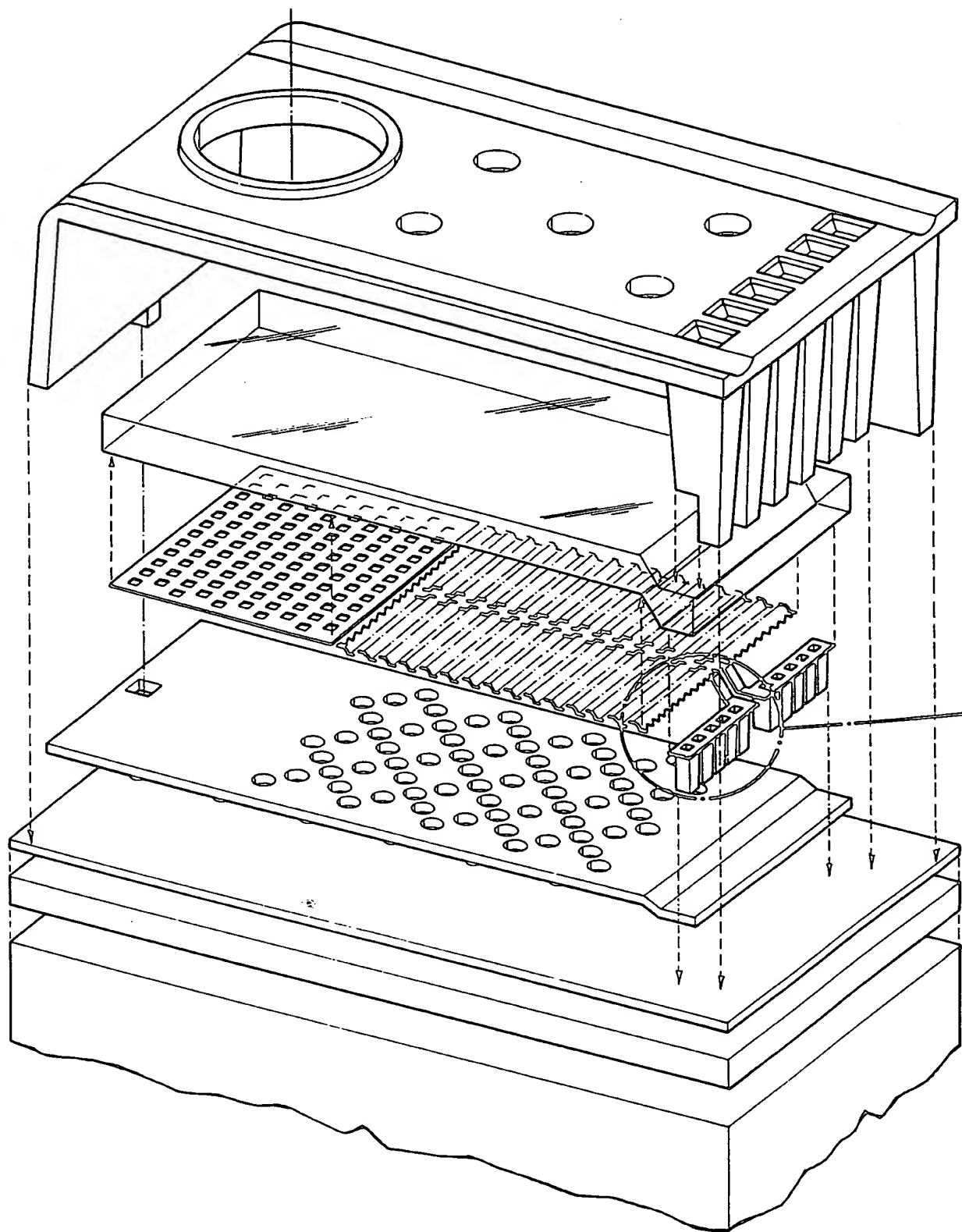


Fig. C30.2

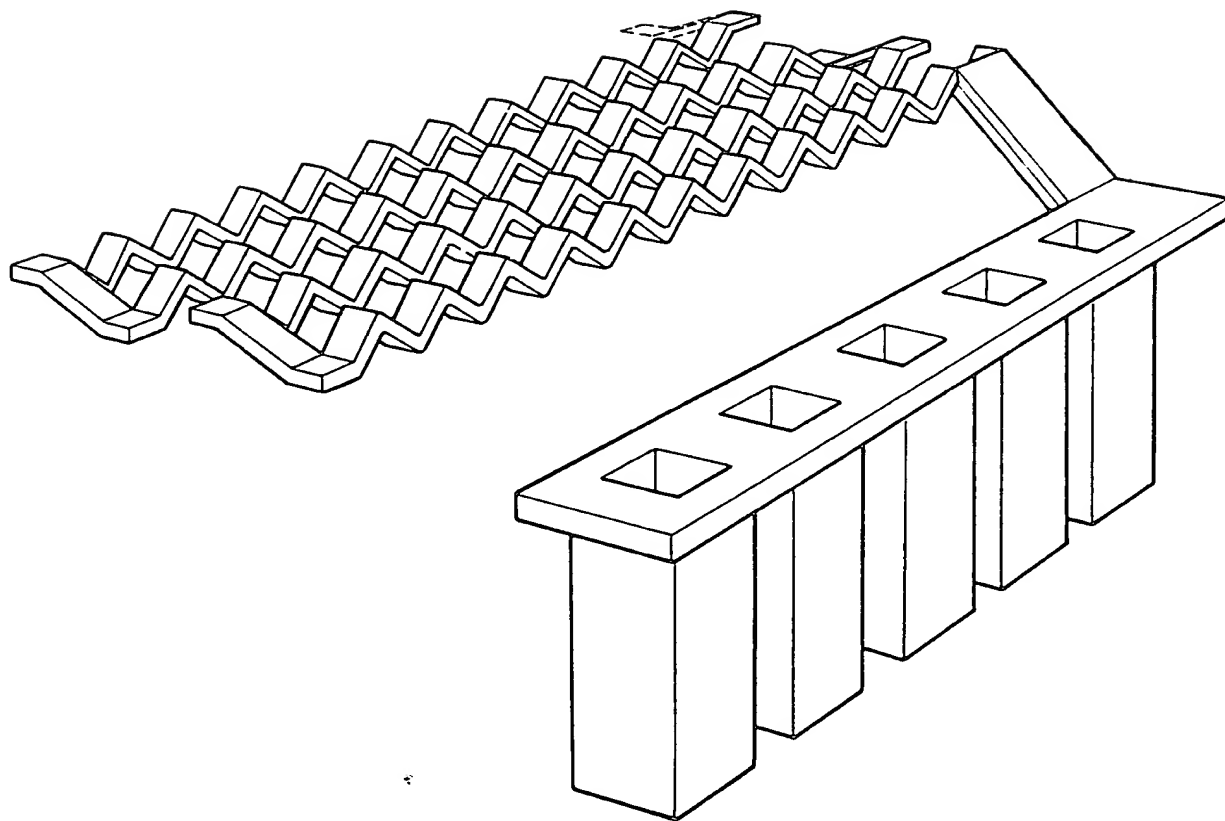


Fig. C30.3

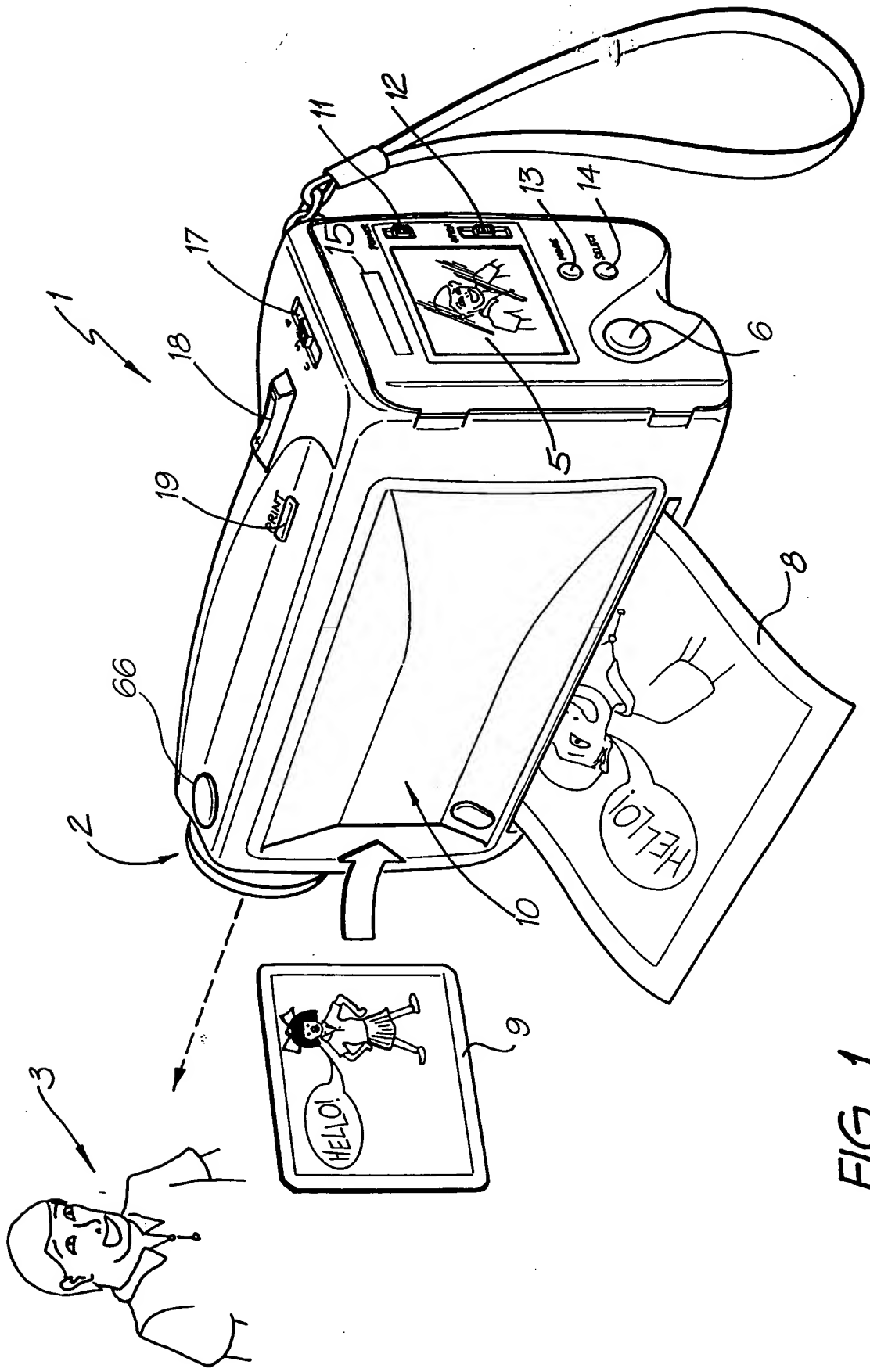


FIG. 1

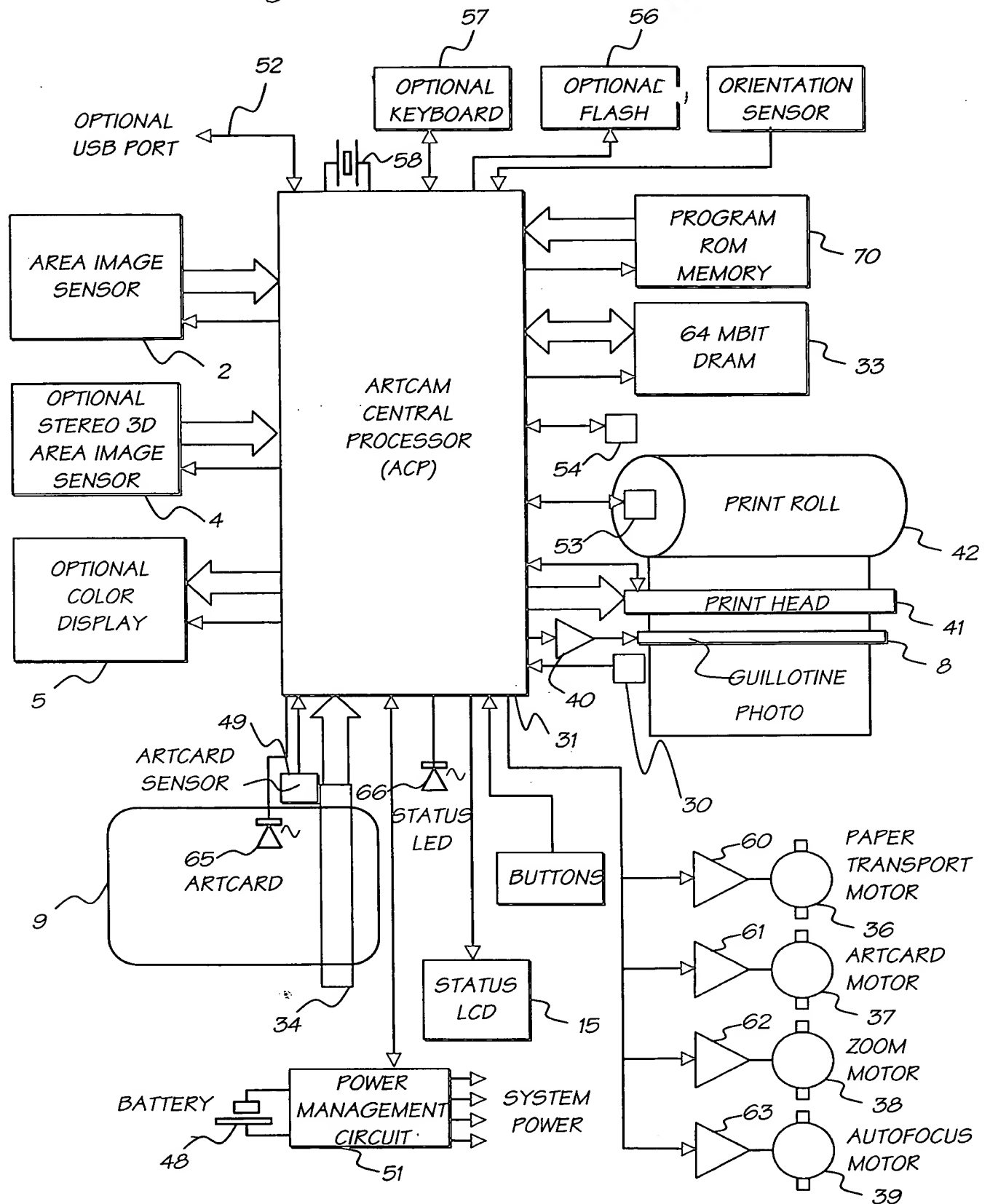


FIG. 2

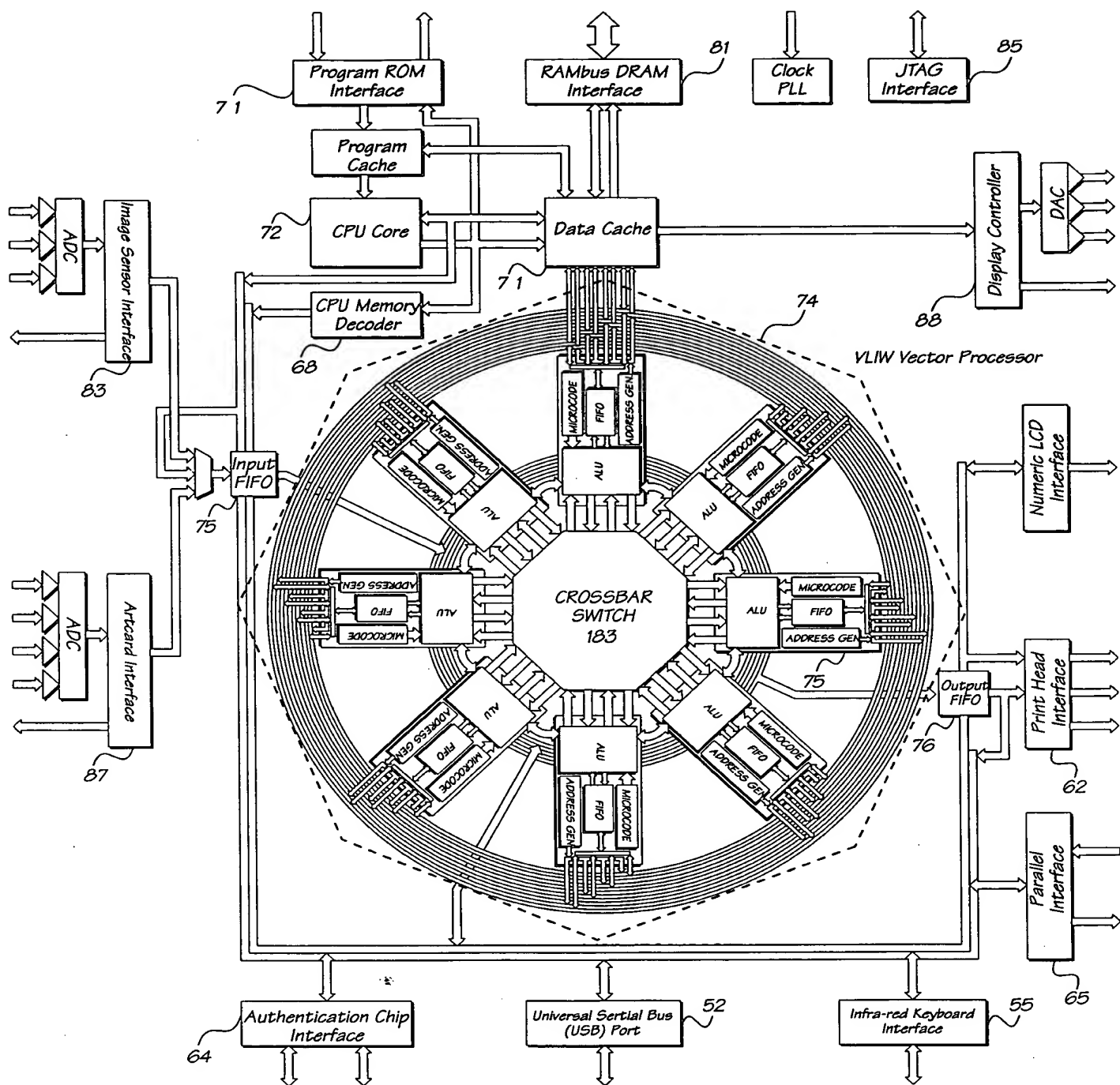
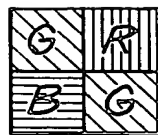


FIG. 3



2x2 PIXEL BLOCK FROM CCD

FIG. 4

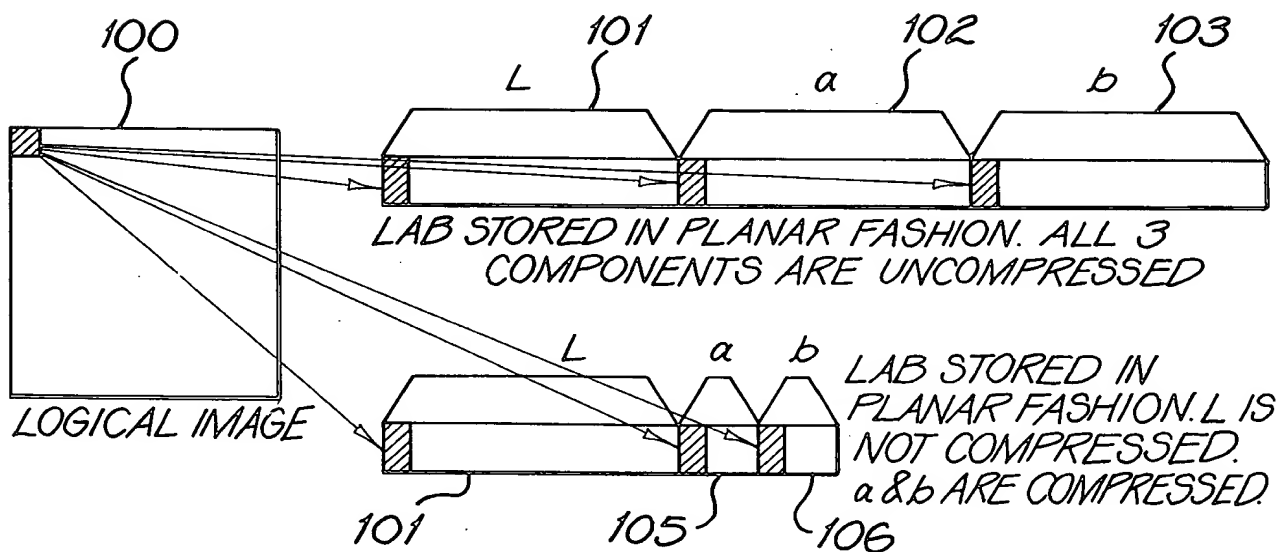


FIG. 5

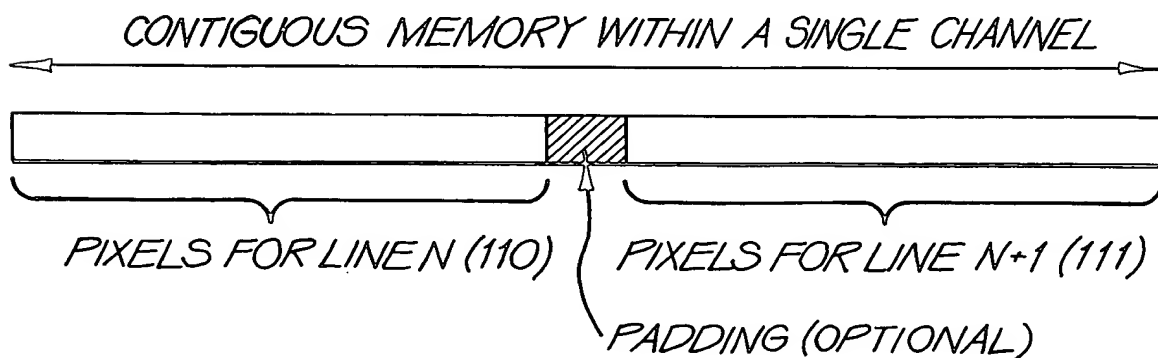


FIG. 6

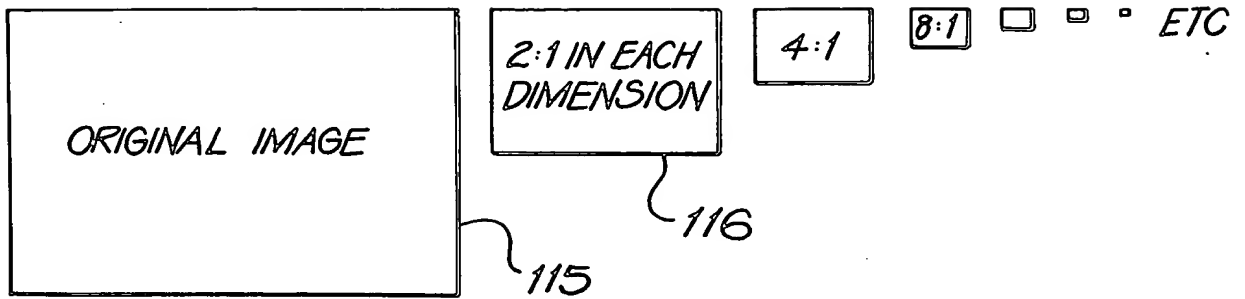


FIG. 7

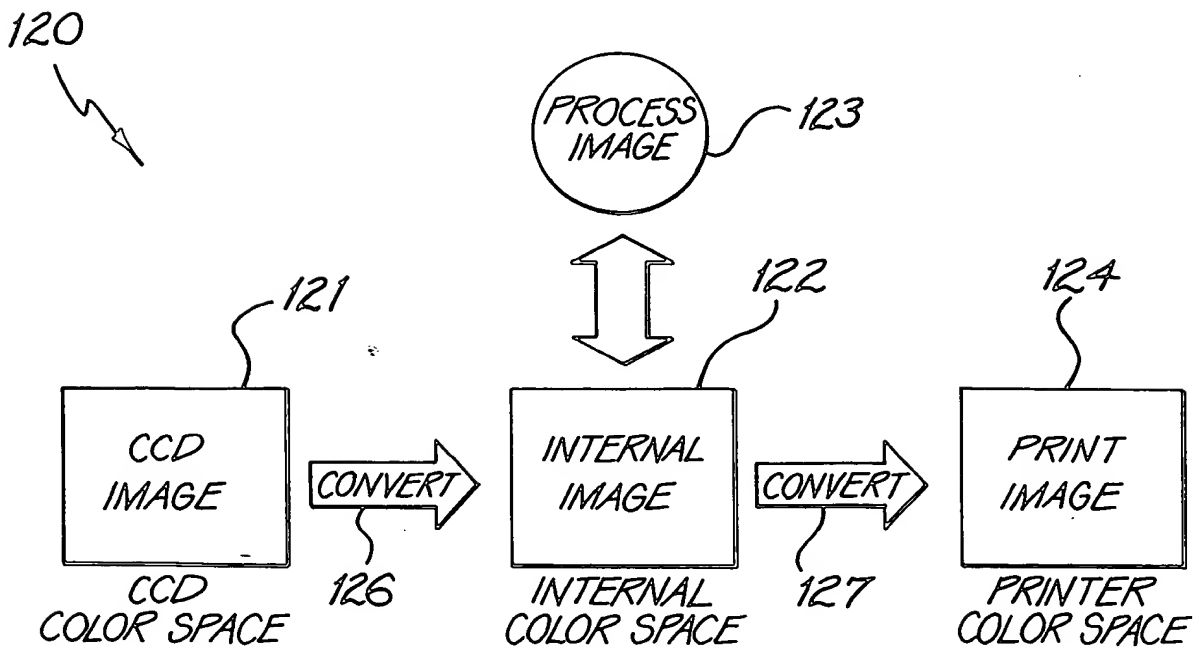


FIG. 8

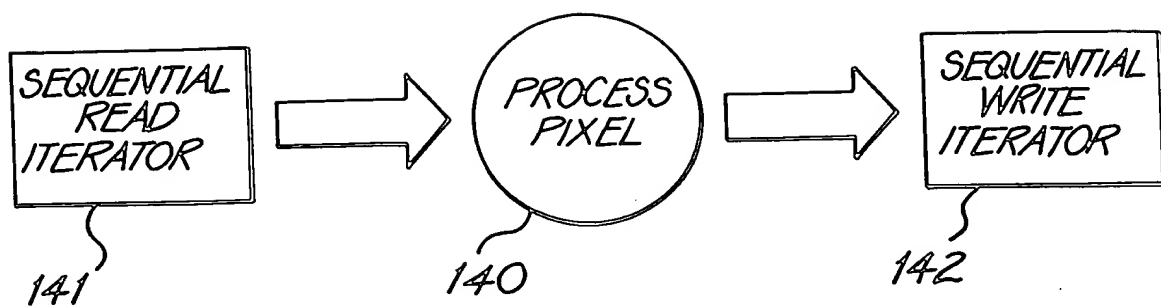


FIG. 9

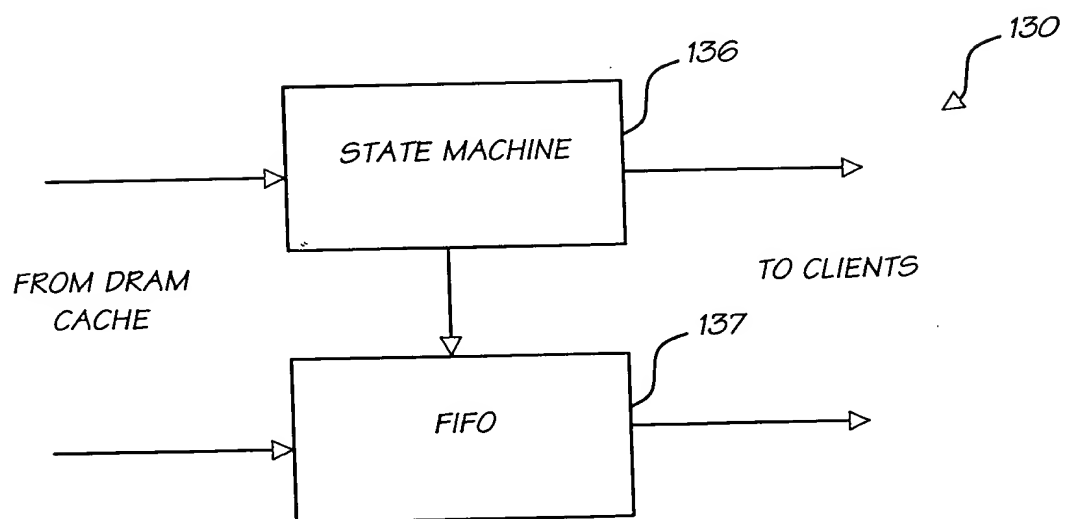


FIG. 10

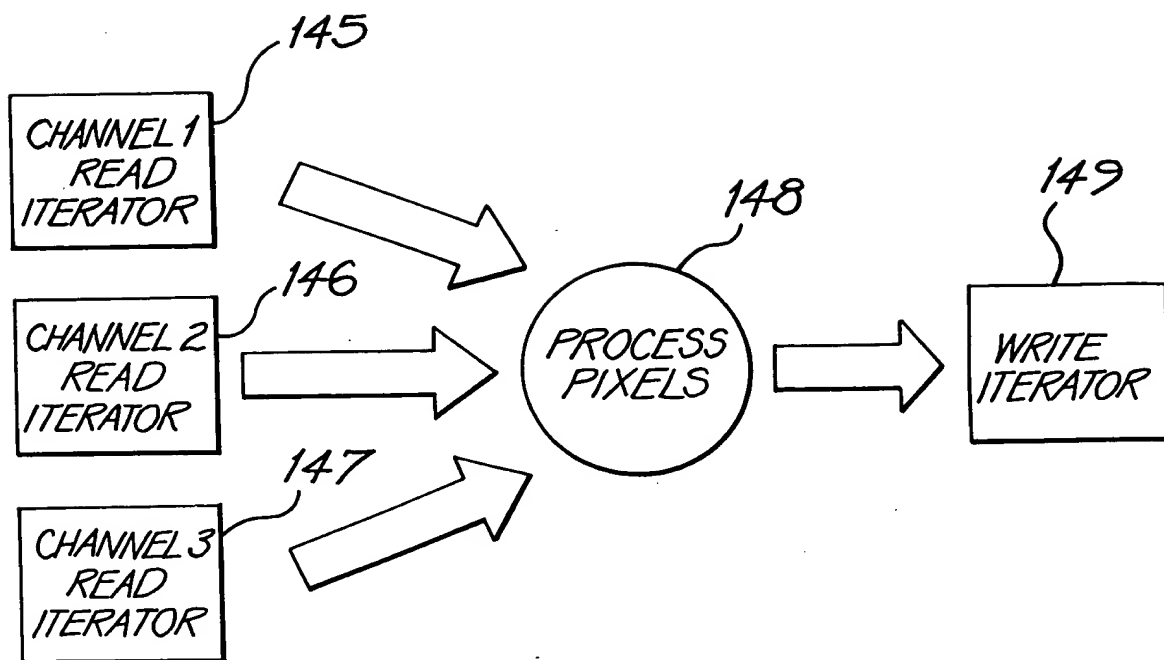


FIG. 11

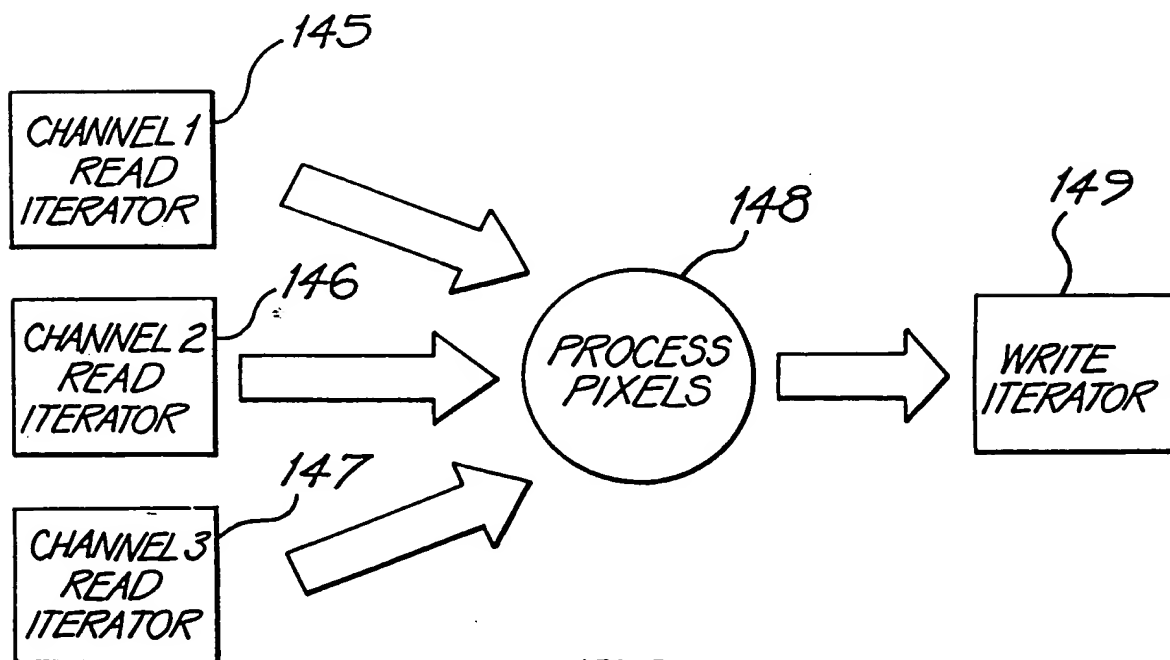
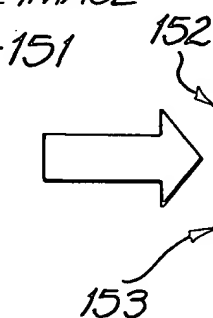
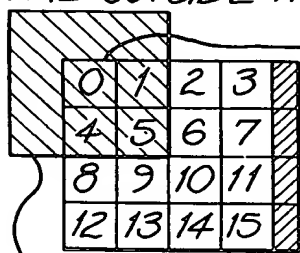


FIG. 12

A 3x3 BOX VIEW TRAVERSES THE PIXELS IN ORDER: 0,1,2,3,4,5,6,7,8 ETC,
PLACING A 3x3 BOX CENTERED OVER EACH PIXEL...

3x3 BOX VIEW OF FIRST PIXEL IN
IMAGE = 9 PIXELS, 5 OF WHICH
ARE OUTSIDE THE IMAGE



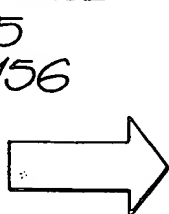
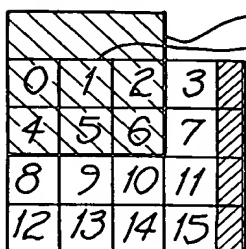
FIRST 9 PIXELS FROM THE
BOX READ ITERATOR:

IF DUPLICATION OF EDGE PIXELS IS ON:
0,0,0,0,0,1,4,4,5

IF DUPLICATION OF EDGE PIXELS IS OFF:
V,V,V,V,0,1,V,4,5
WHERE V IS CONSTANT
"OUTSIDE IMAGE" PIXEL VALUE

FIG. 13

3x3 BOX VIEW OF SECOND PIXEL IN
IMAGE = 9 PIXELS, 3 OF WHICH
ARE OUTSIDE THE IMAGE



SECOND 9 PIXELS FROM THE
BOX READ ITERATOR:

IF DUPLICATION OF EDGE PIXELS IS ON:
0,1,2,0,1,2,4,5,6

IF DUPLICATION OF EDGE PIXELS IS OFF:
V,V,V,0,1,2,4,5,6
WHERE V IS CONSTANT
"OUTSIDE IMAGE" PIXEL VALUE

FIG. 14

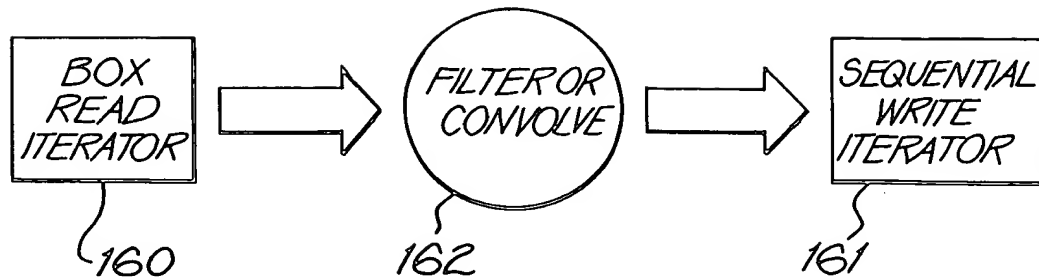
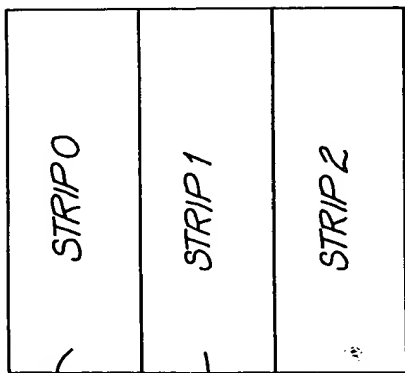
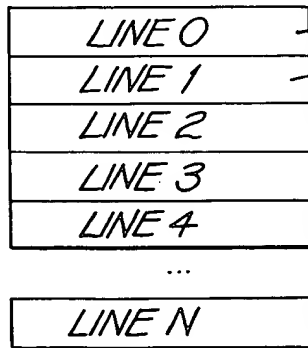


FIG. 15

IMAGE BROKEN INTO VERTICAL STRIPS, EACH STRIP IS 32 PIXELS ACROSS.



LINES ARE ACCESSED LINE 0 TO LINE N WITHIN A SINGLE STRIP.



PIXELS ARE ACCESSED PIXEL 0-PIXEL 31 WITHIN A SINGLE LINE.



169

170

167

165

FIG. 16

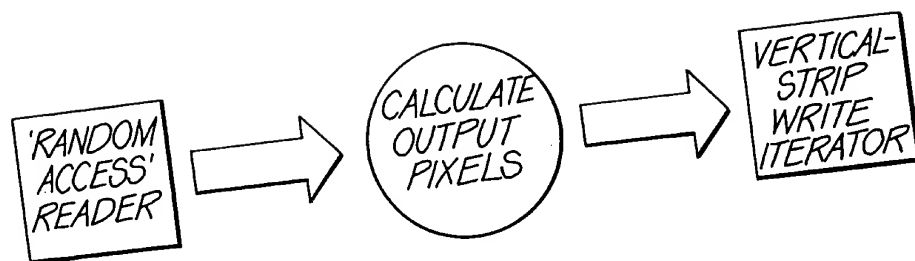


FIG. 17

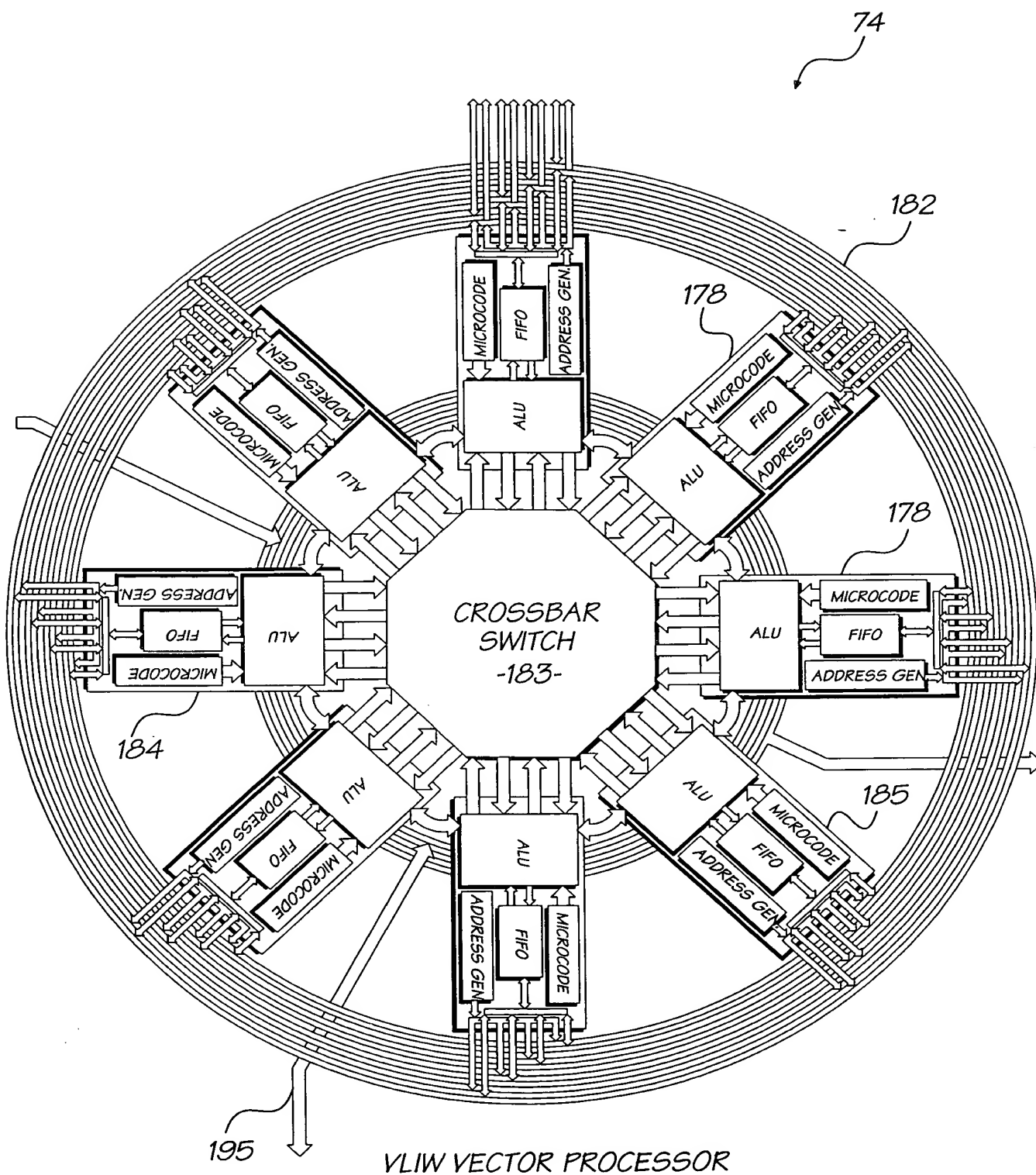


FIG. 18

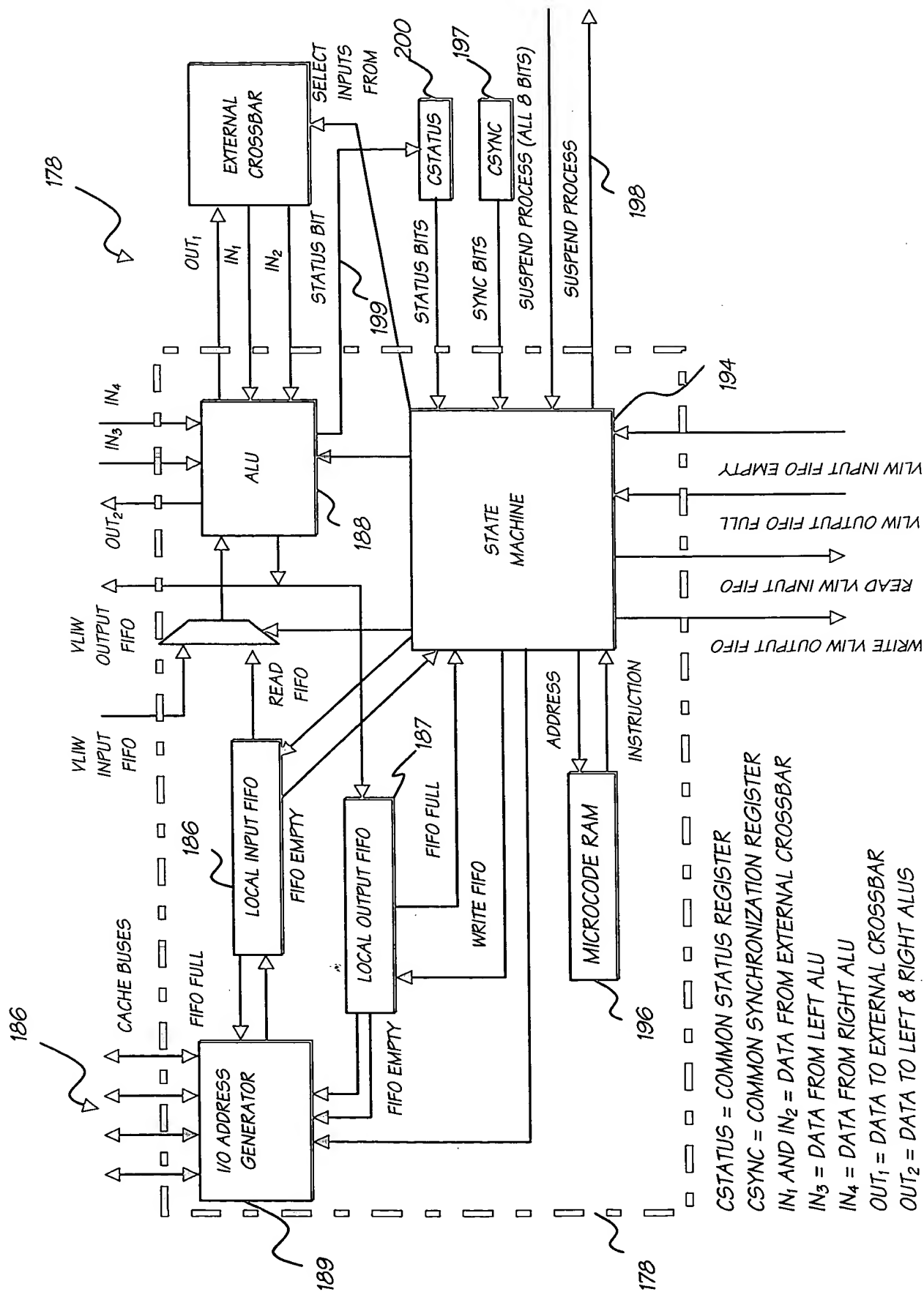


FIG. 19

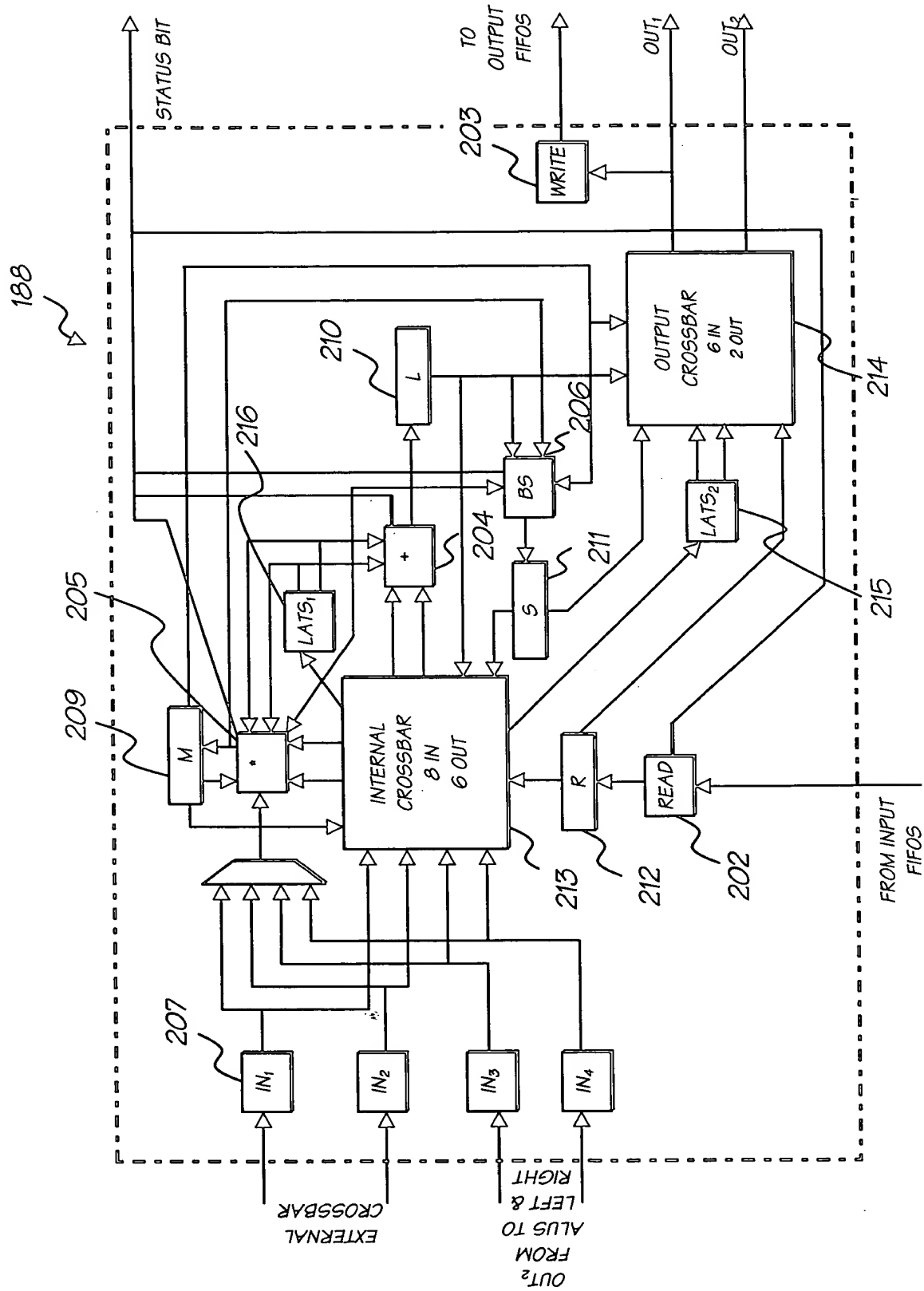


FIG. 20

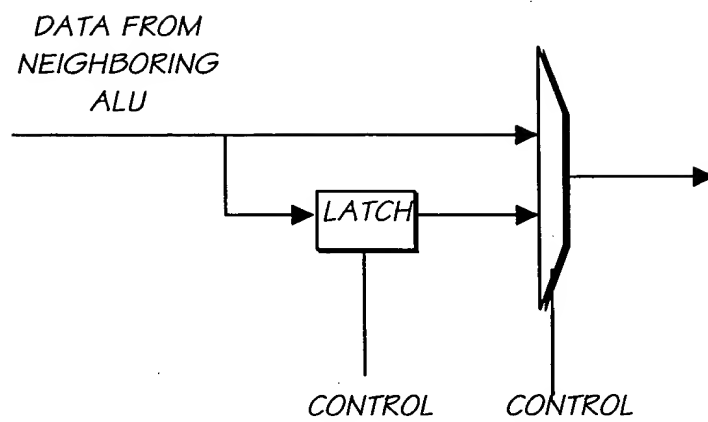


FIG. 21

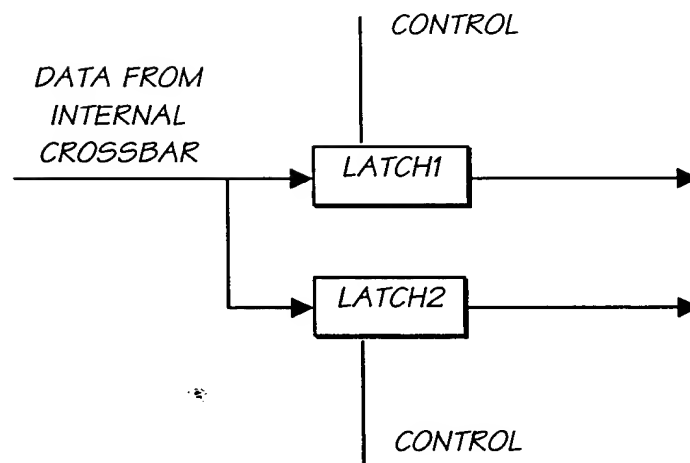


FIG. 22

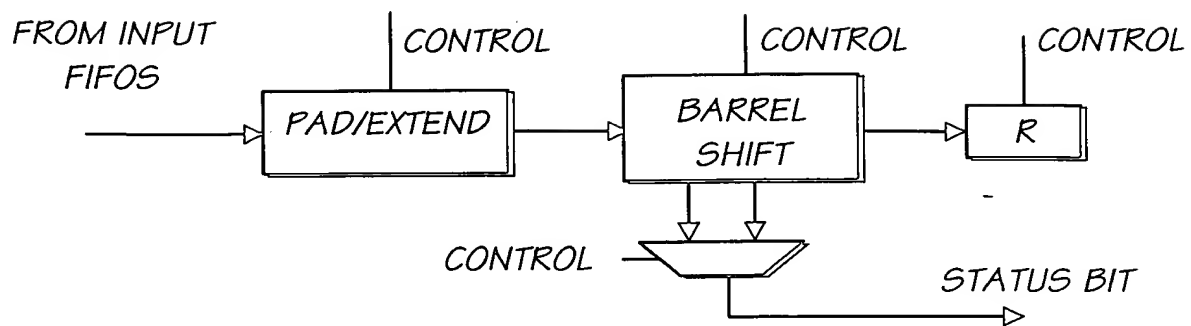


FIG. 23

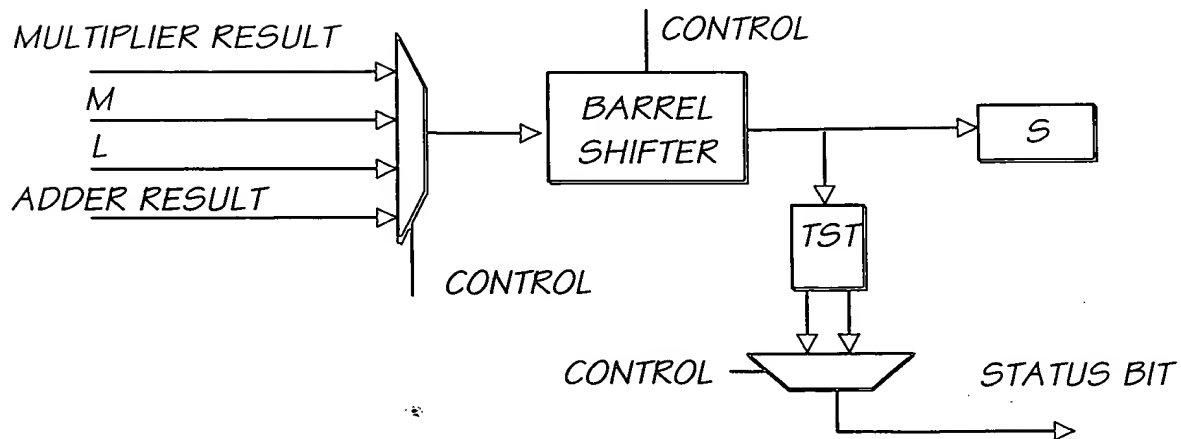


FIG. 24

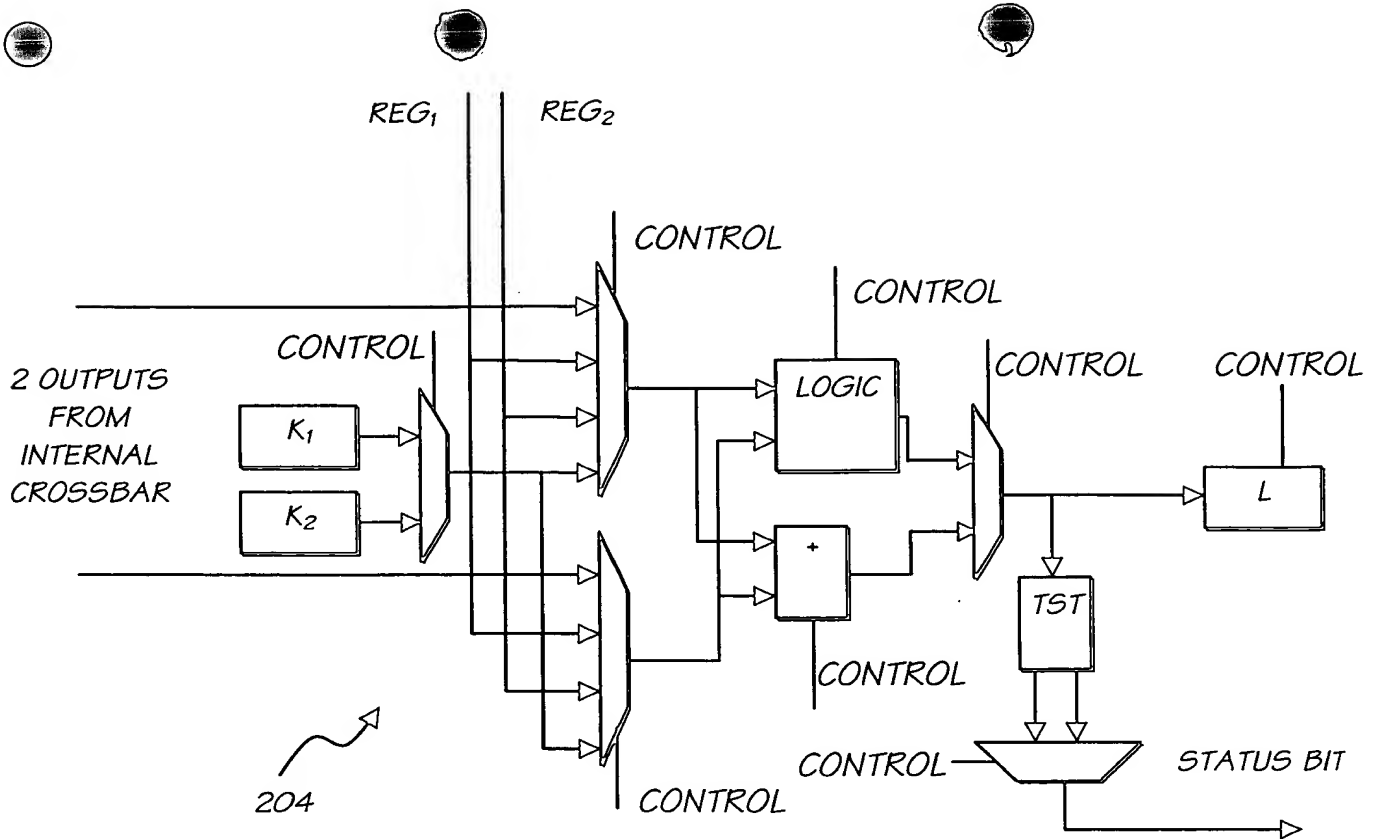


FIG. 25

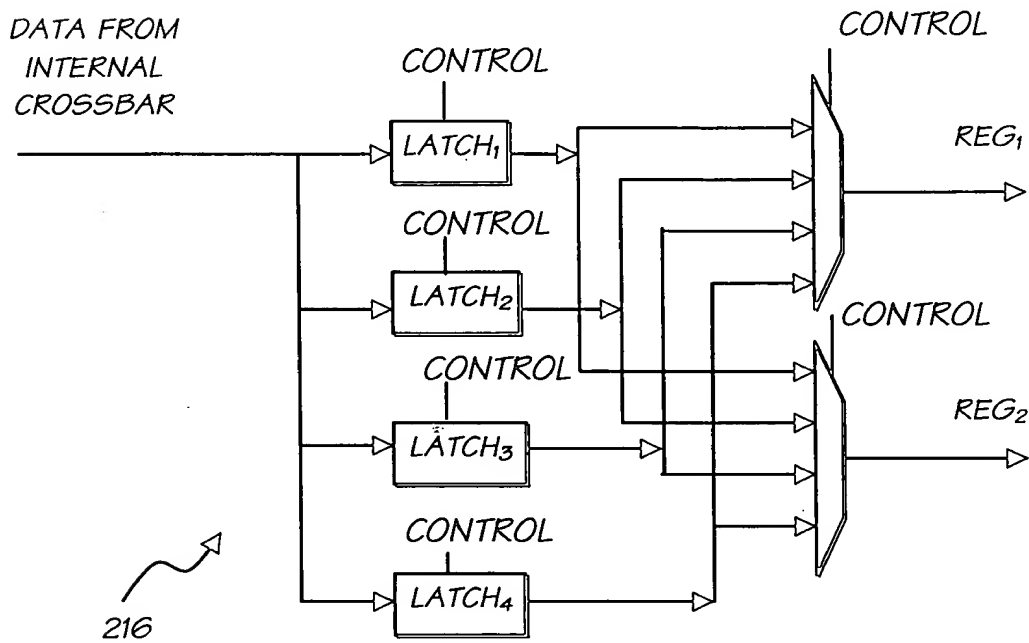


FIG. 26

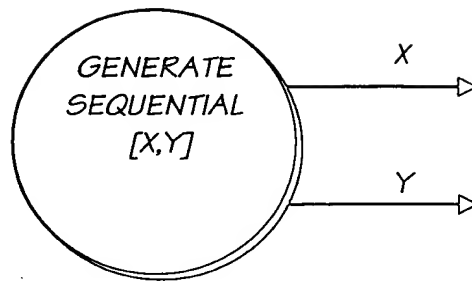


FIG. 28

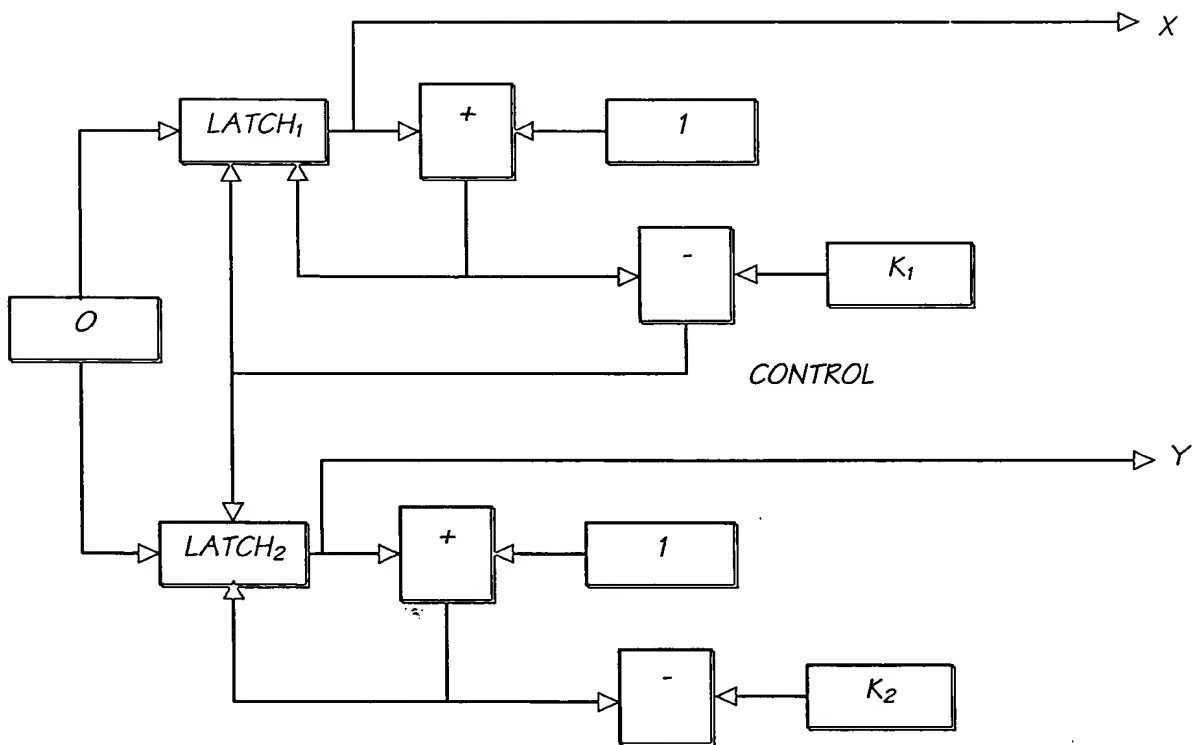


FIG. 29



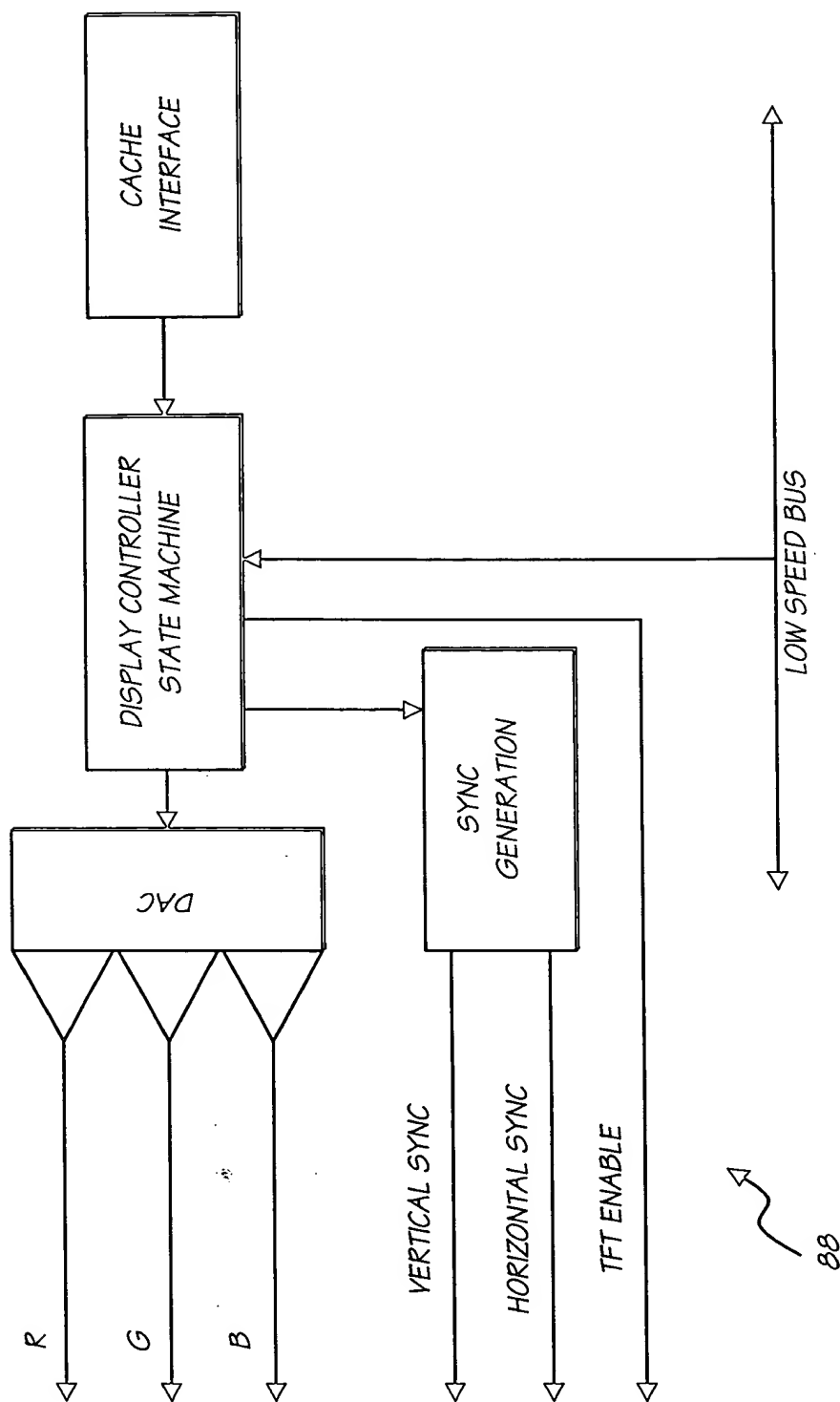


FIG. 32

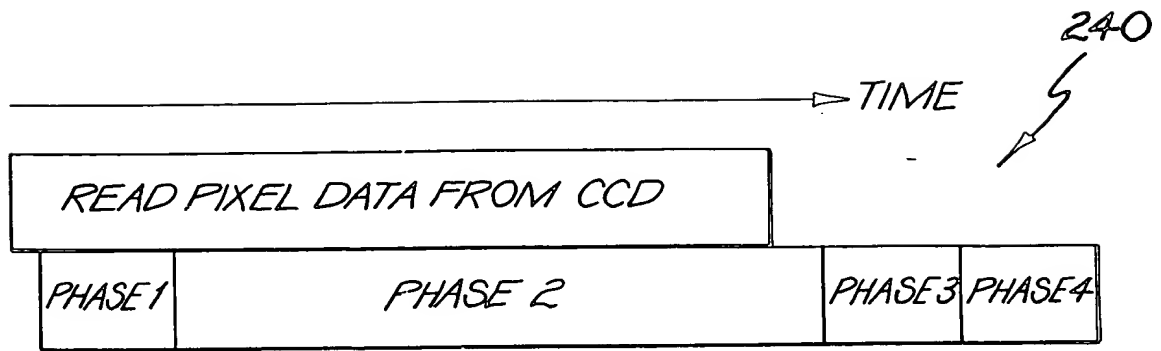


FIG. 33

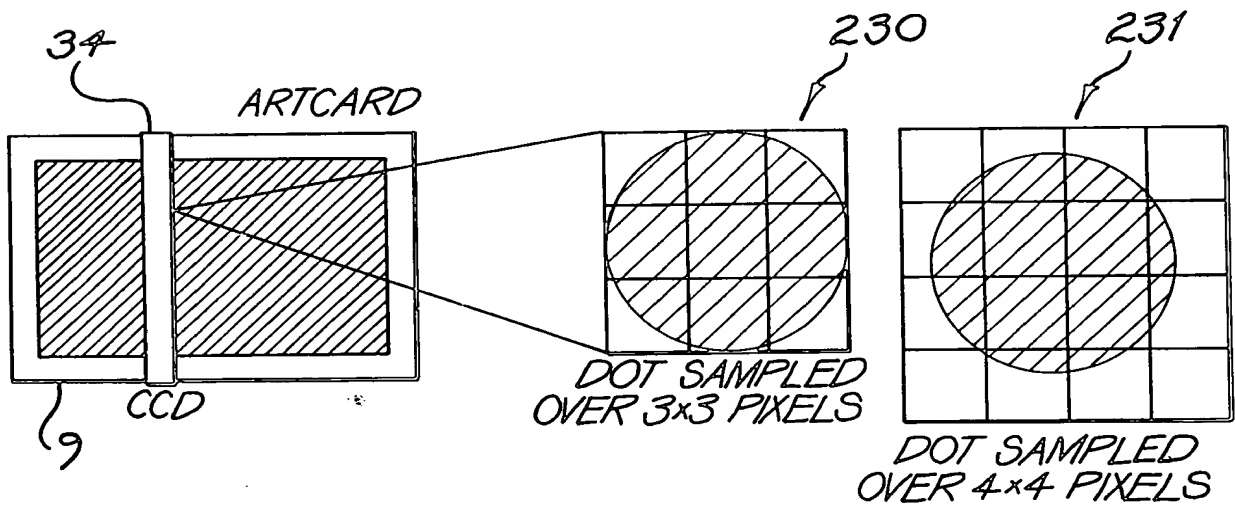


FIG. 34

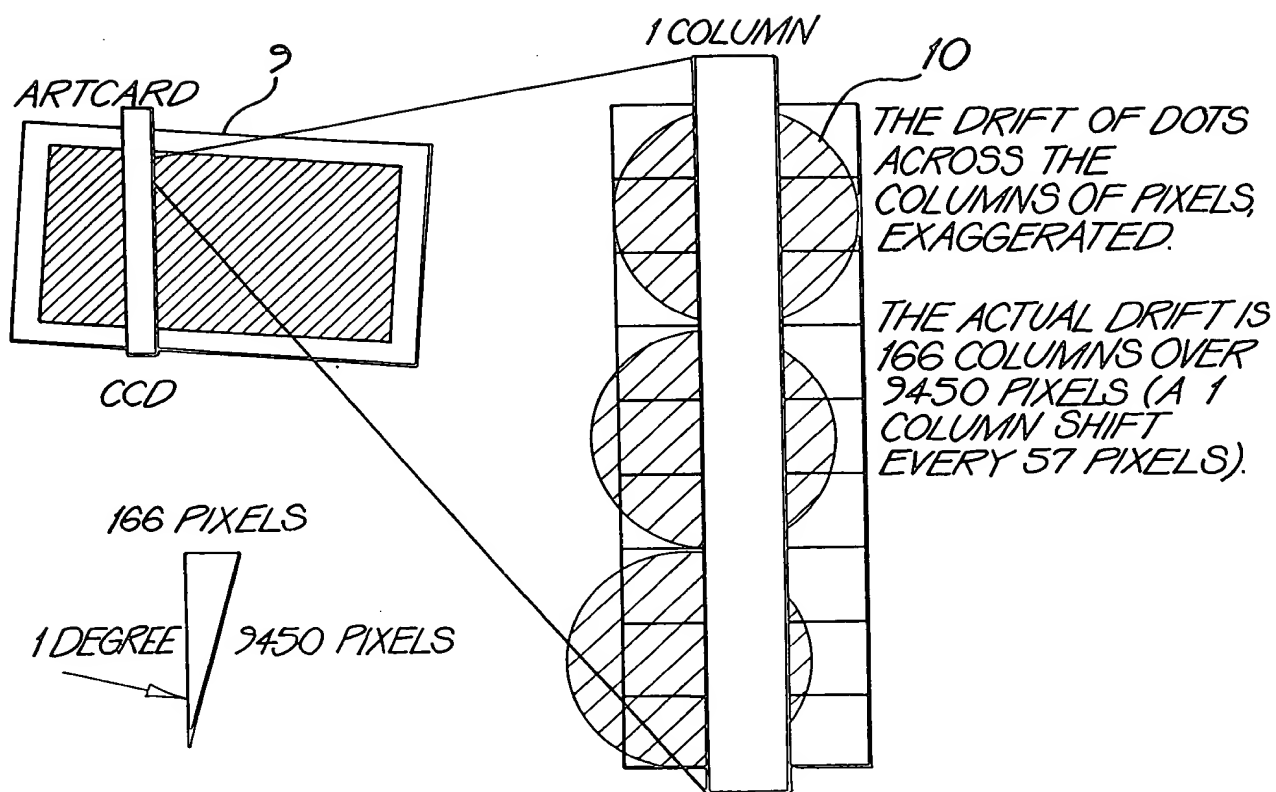


FIG. 35

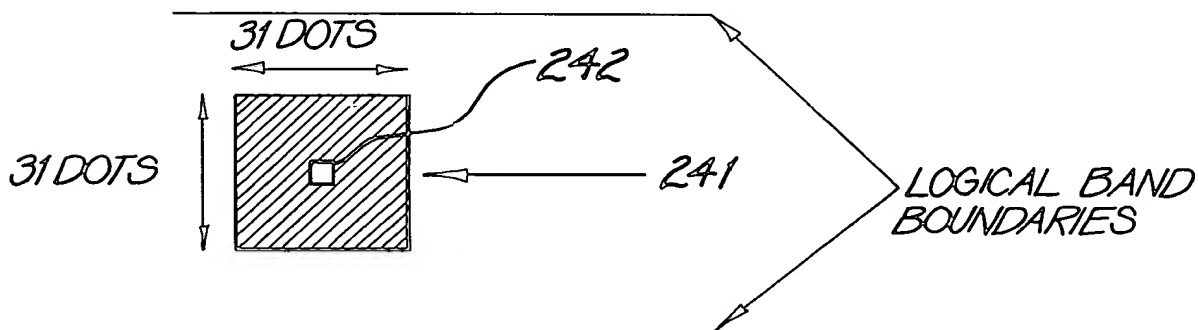


FIG. 38

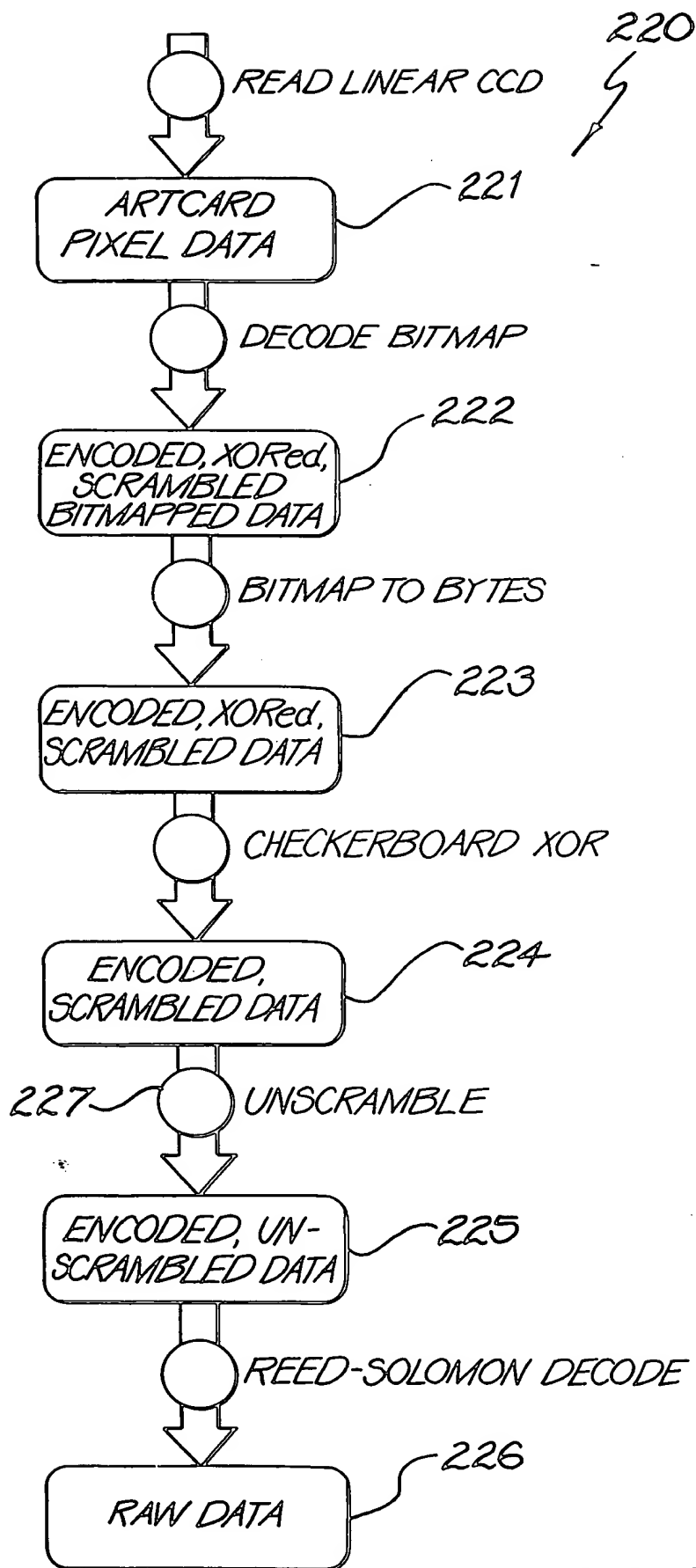


FIG. 36

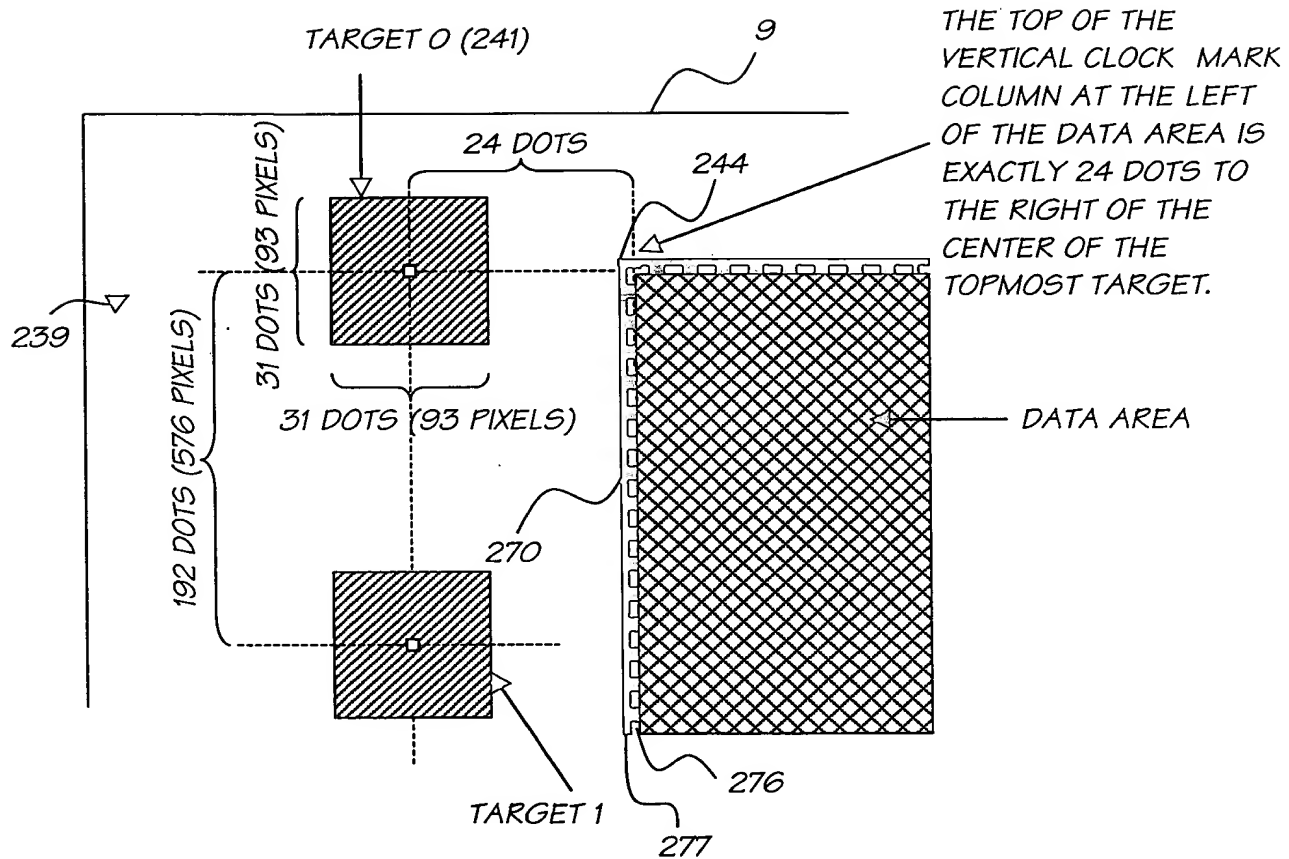


FIG. 37

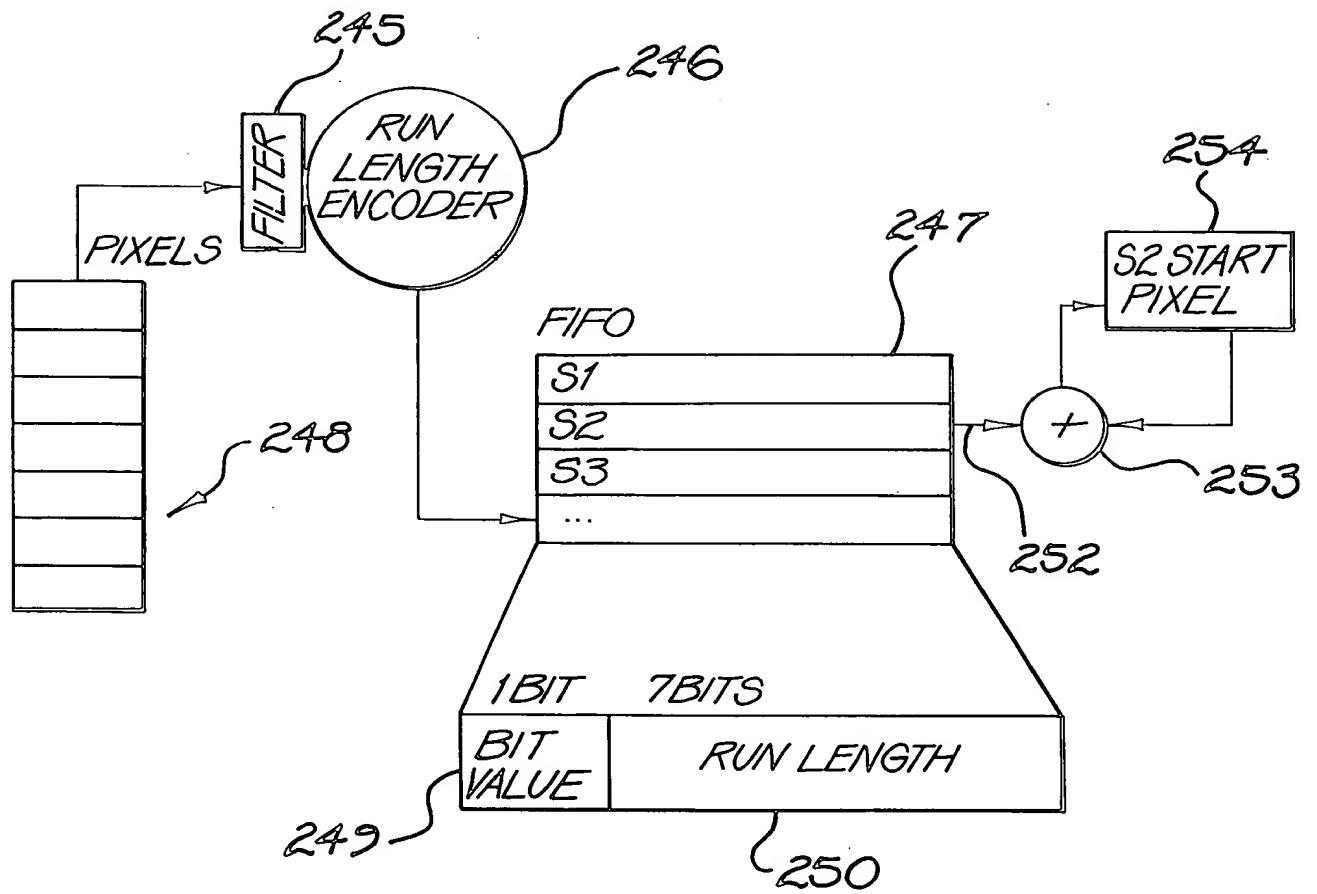


FIG. 39

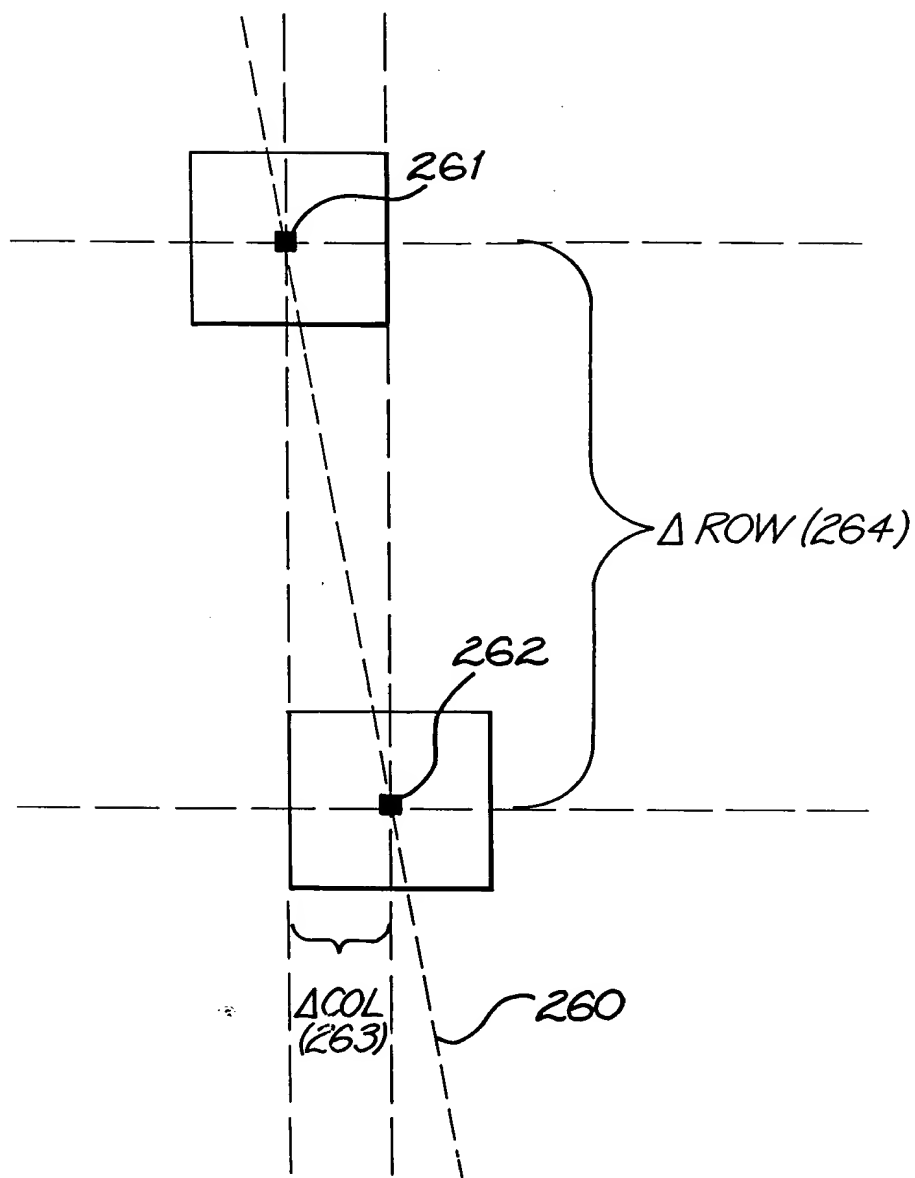


FIG. 40

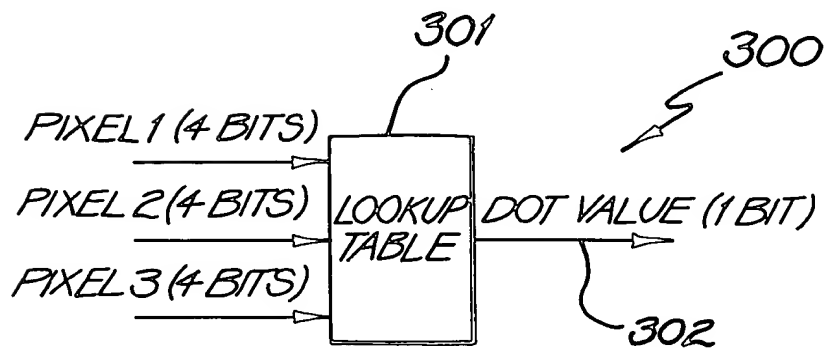
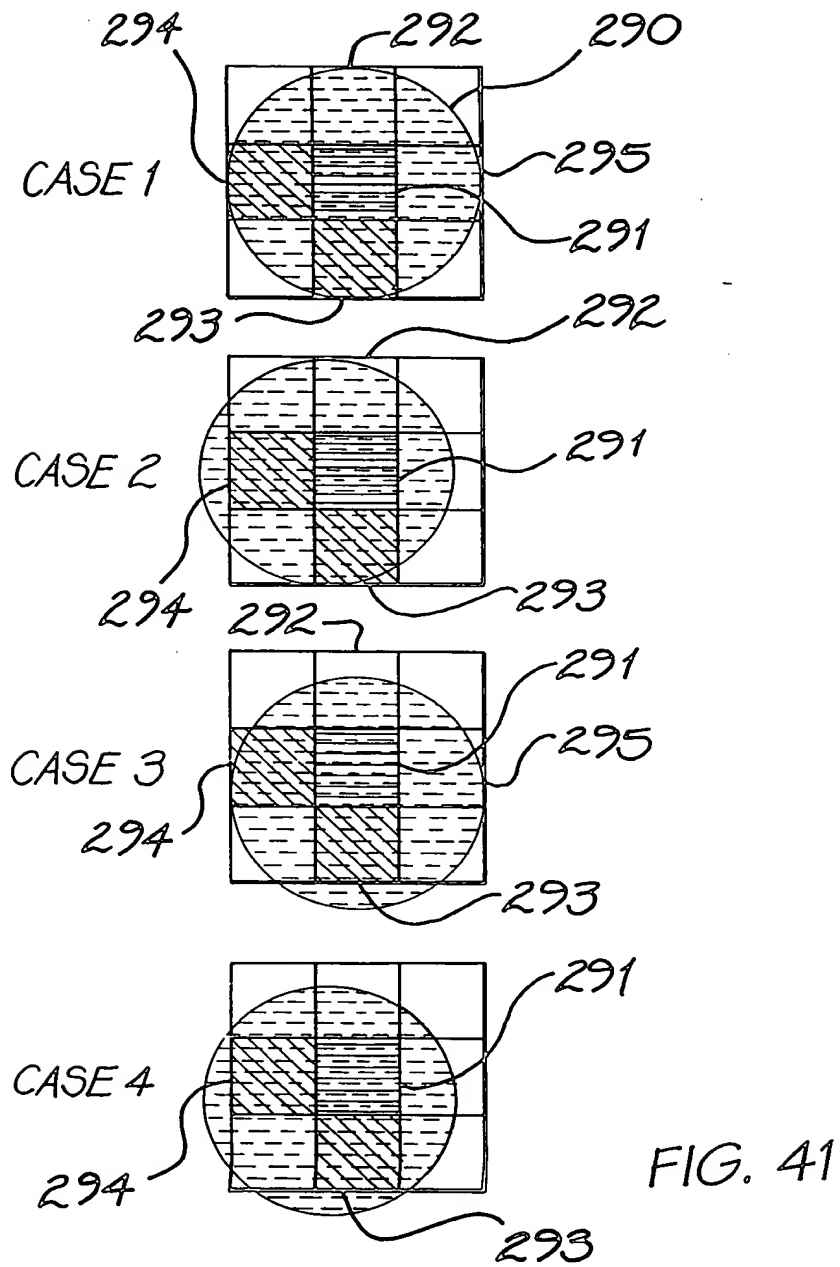


FIG. 42

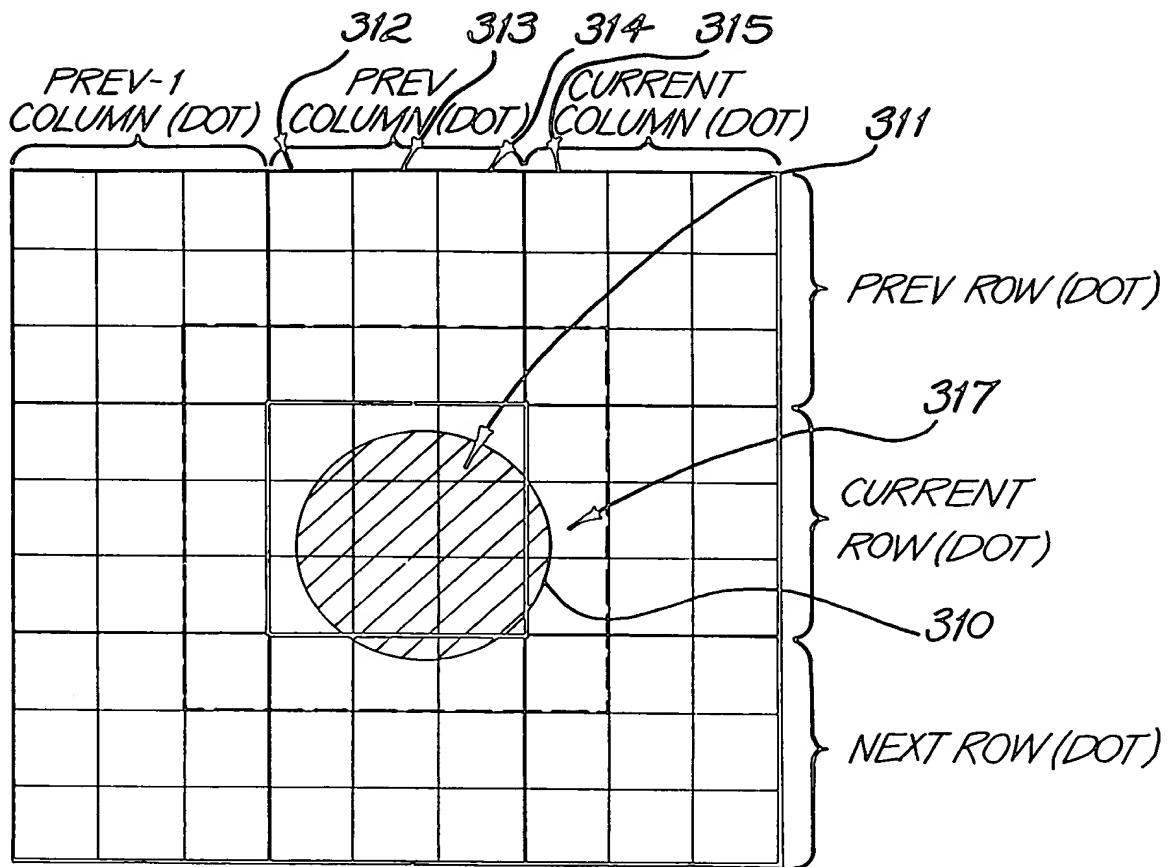


FIG. 43

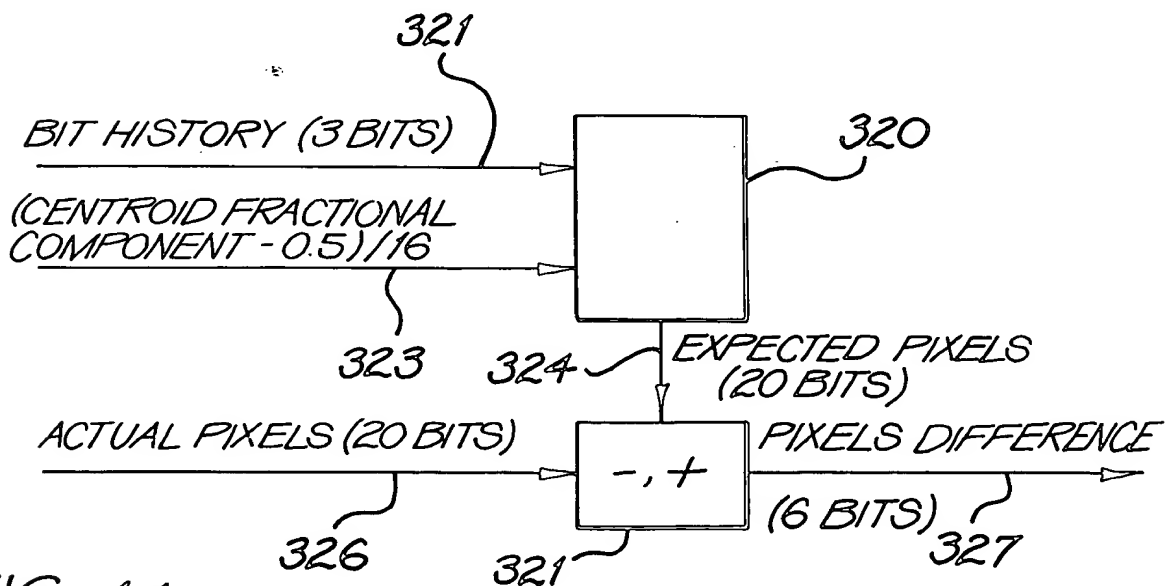


FIG. 44

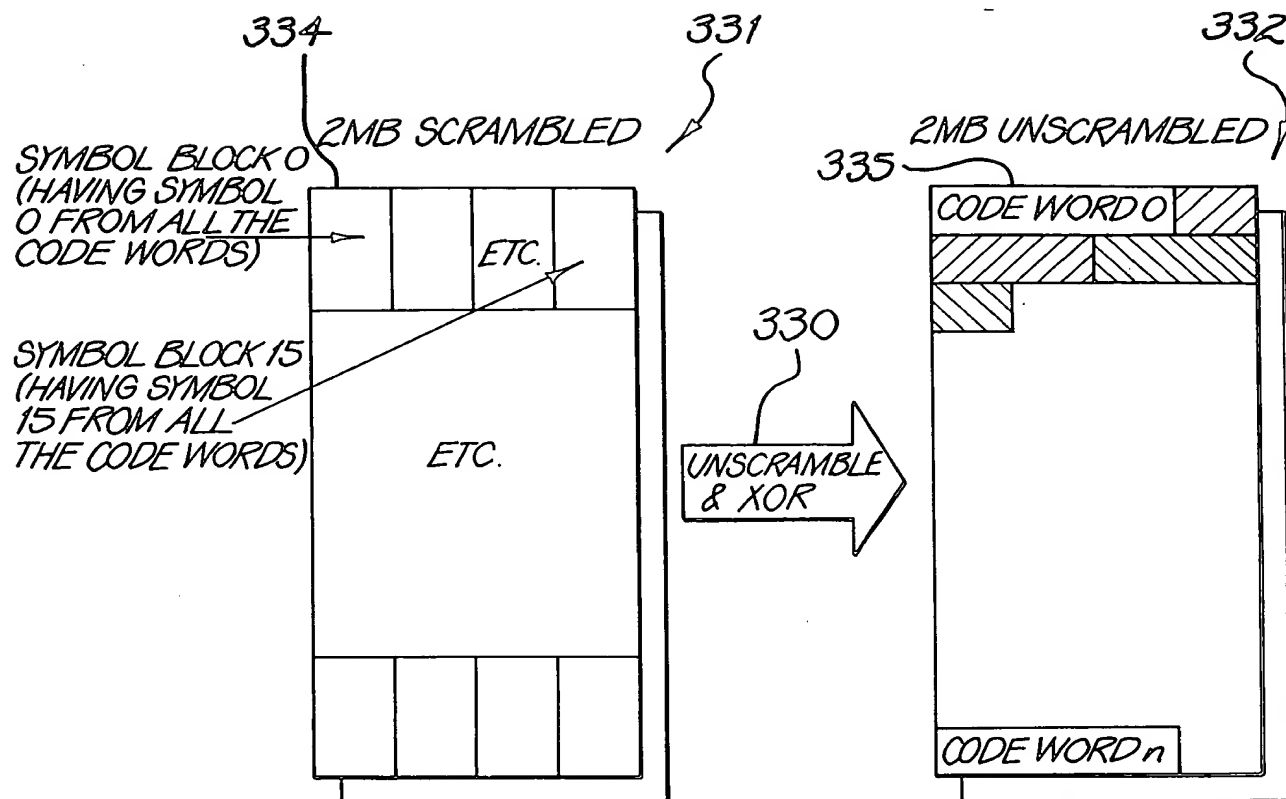


FIG. 45

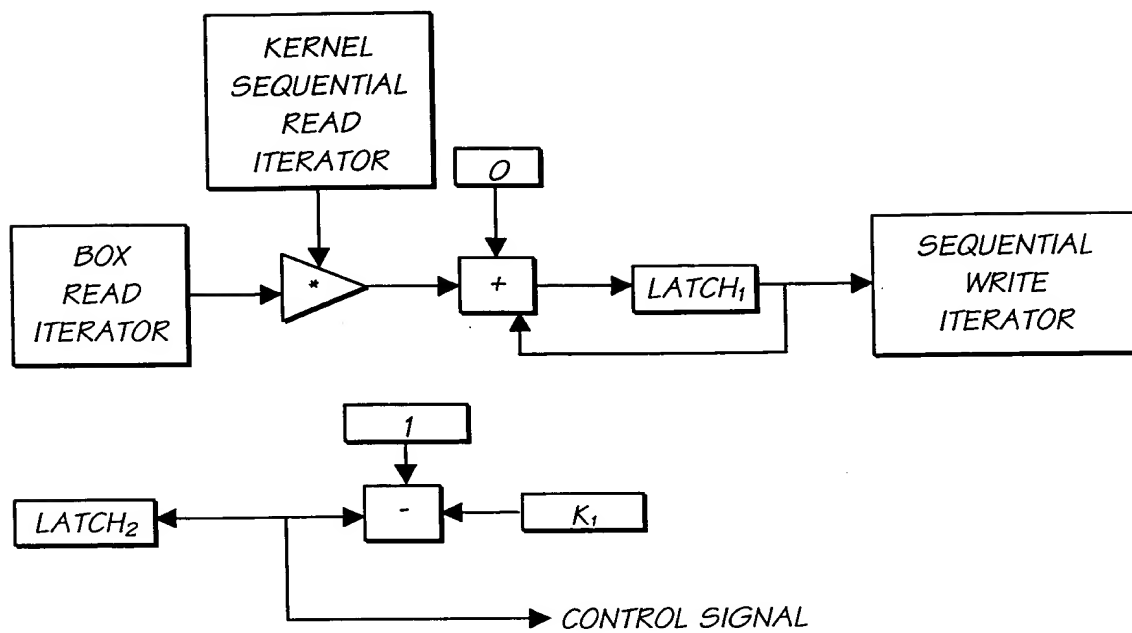


FIG. 46

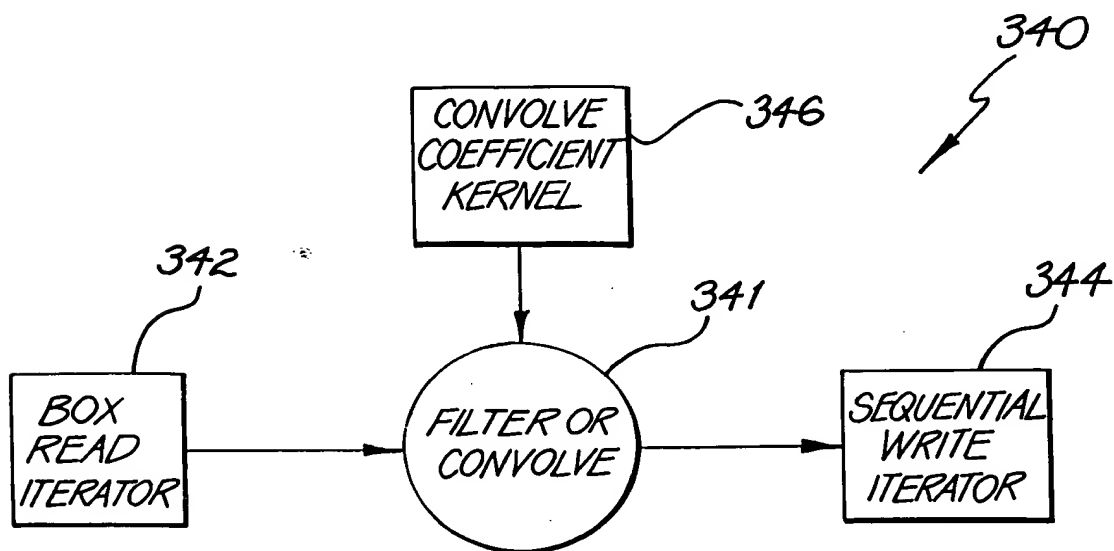


FIG. 47

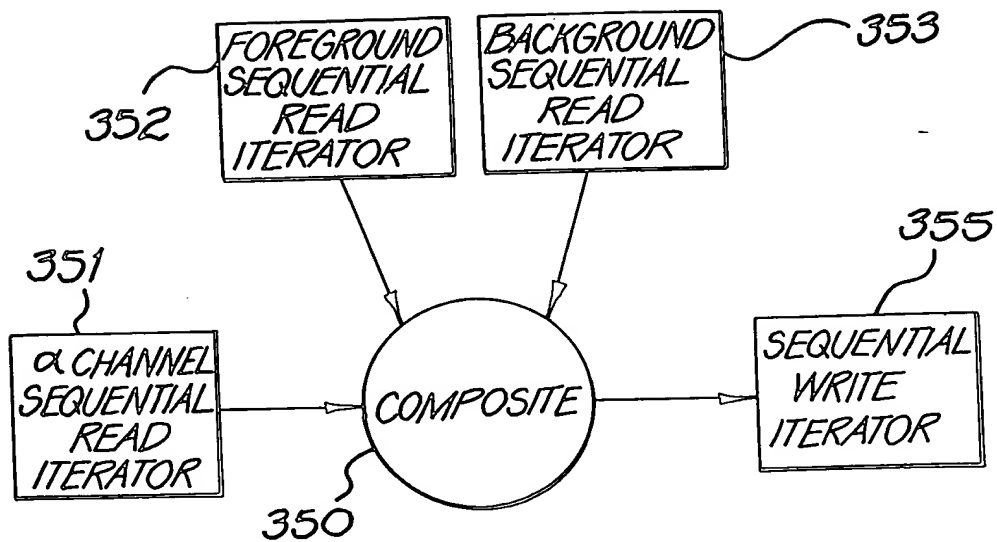


FIG. 48

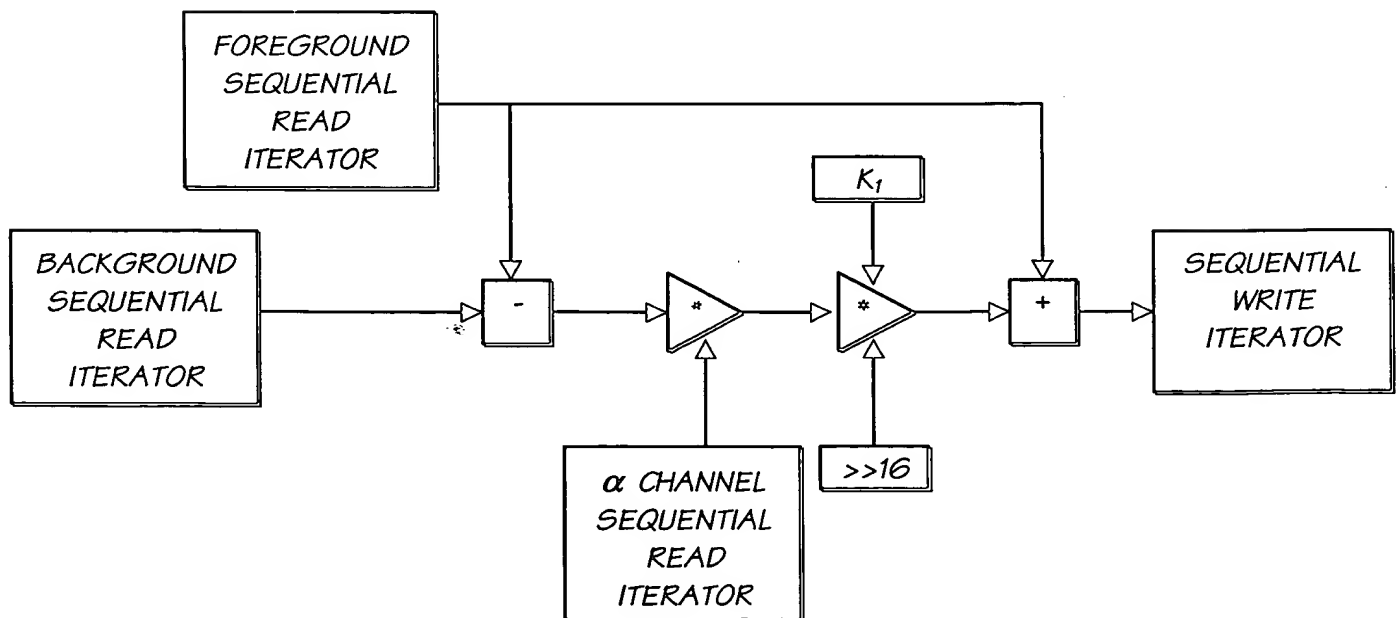


FIG. 49

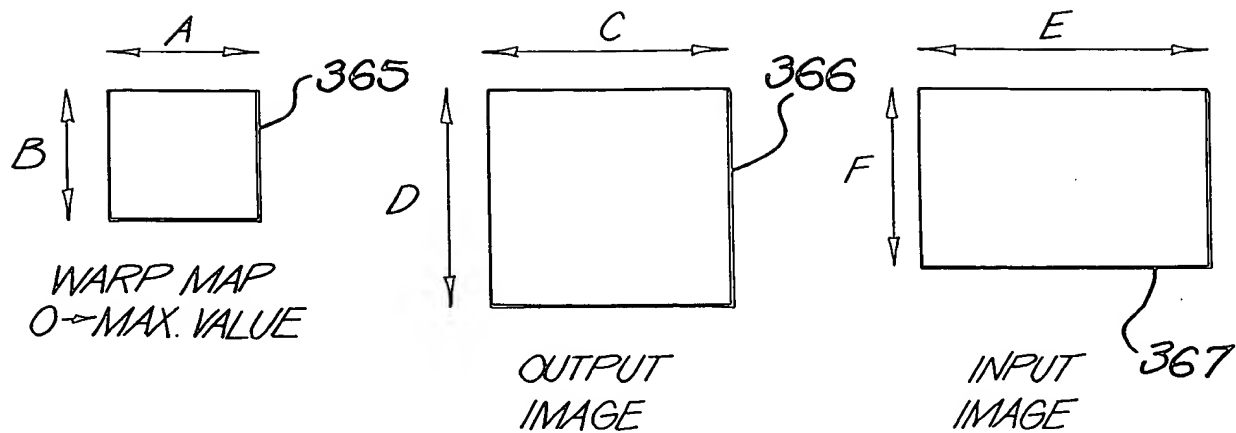


FIG. 50

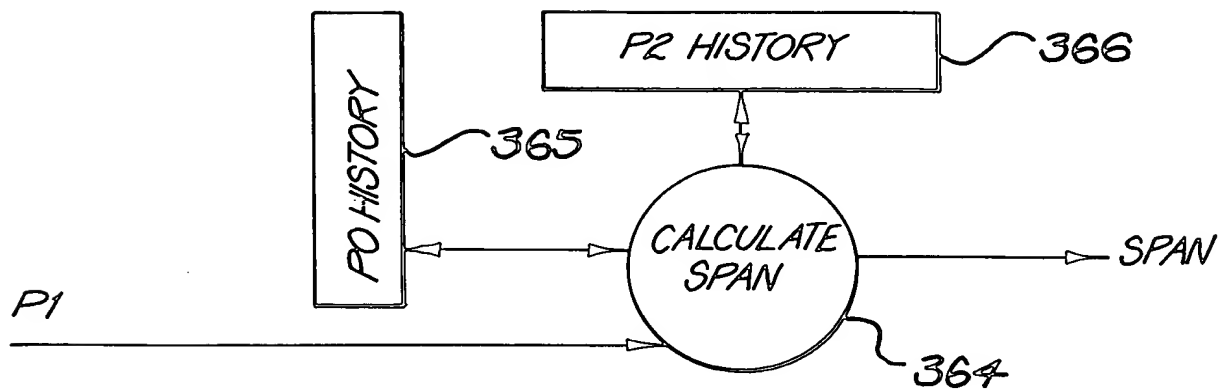


FIG. 53

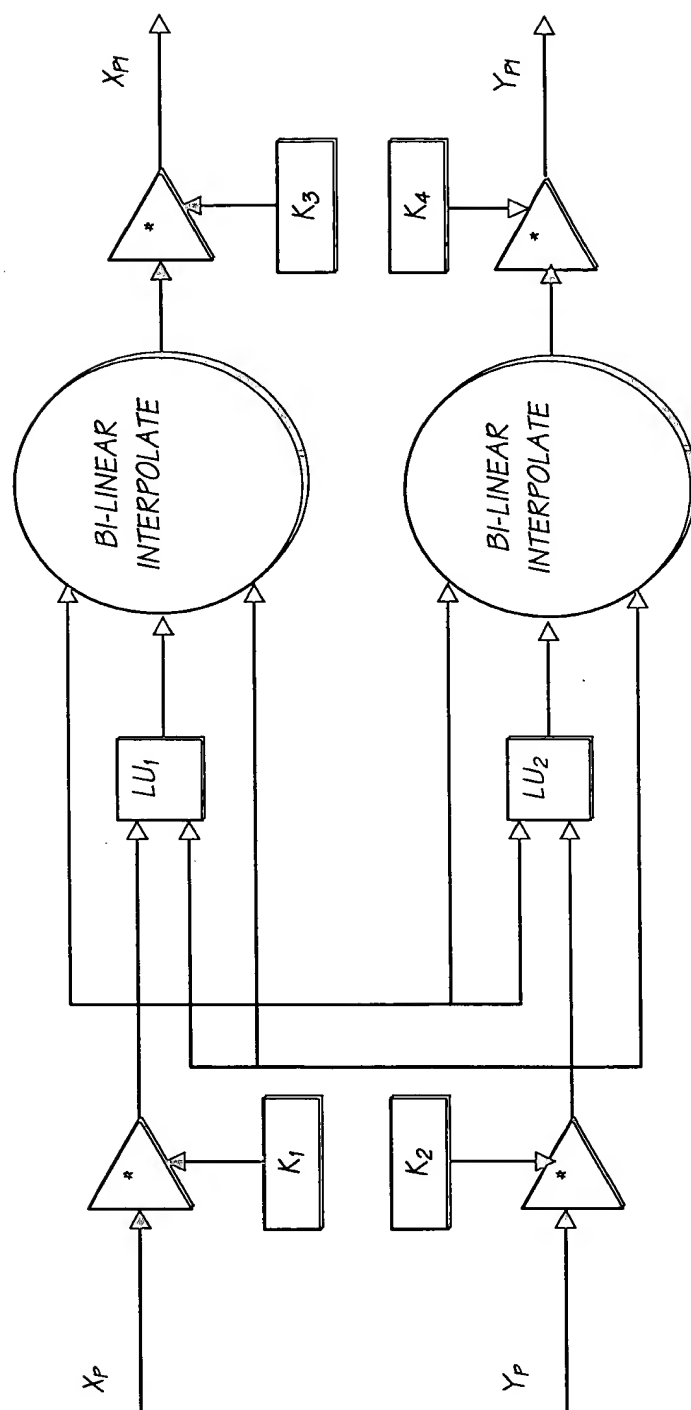


FIG. 51

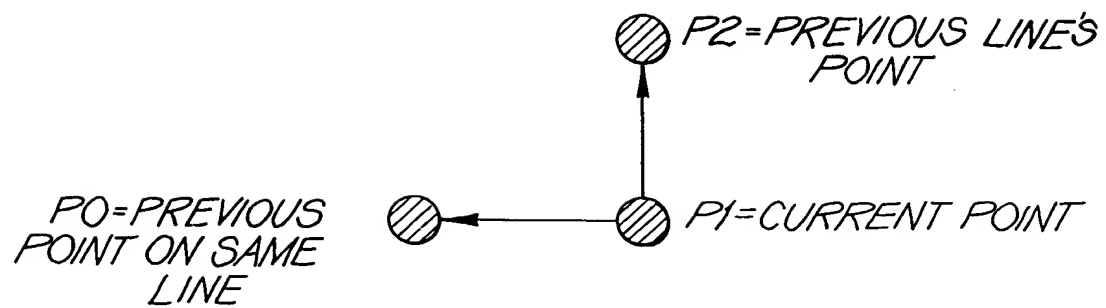


FIG. 52

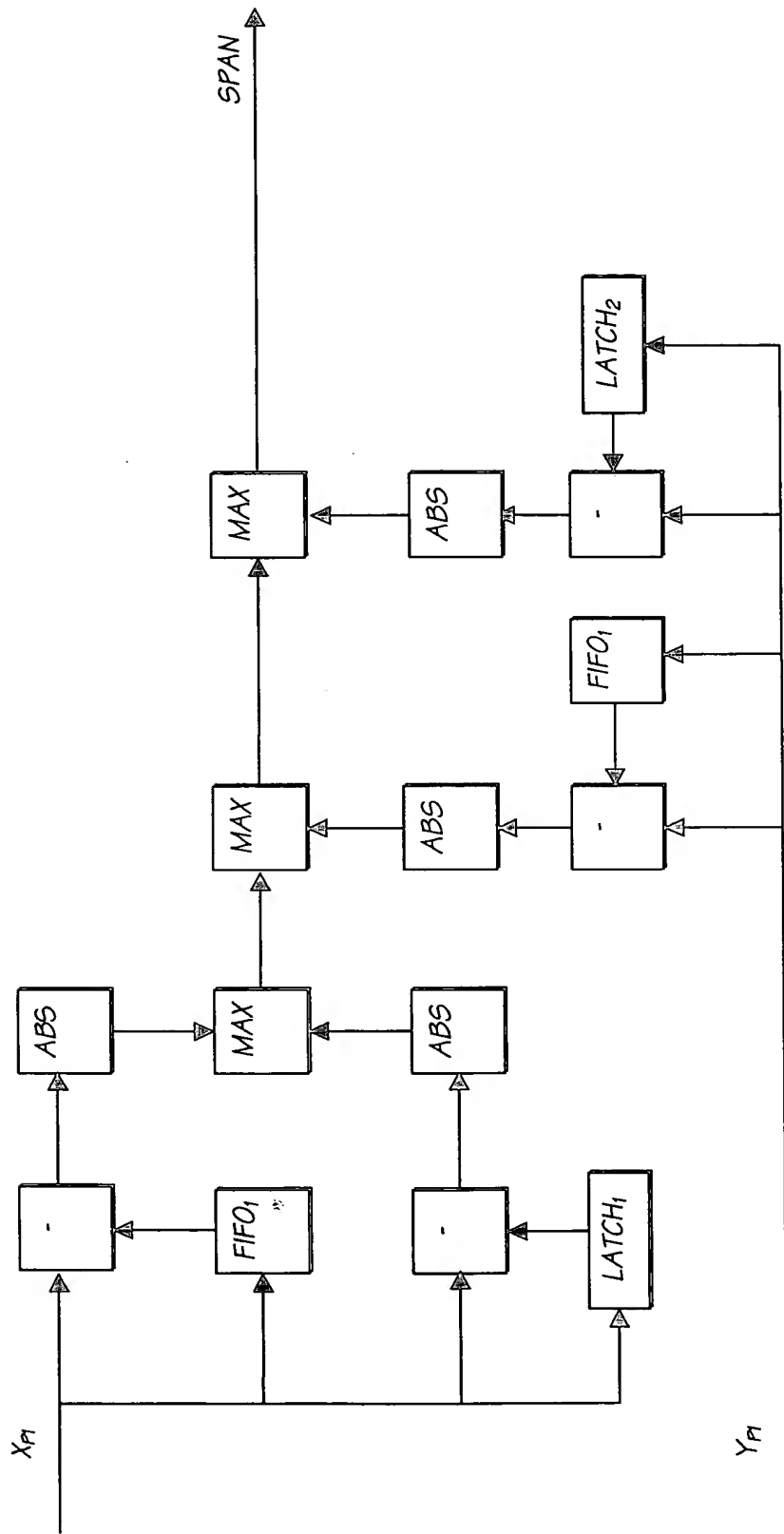


FIG. 54

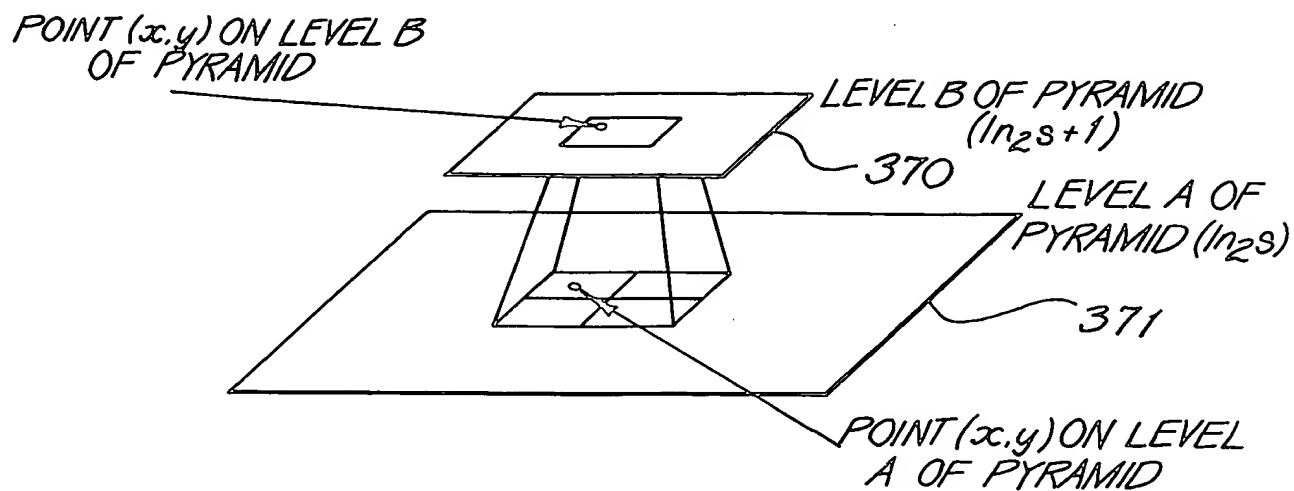


FIG. 55

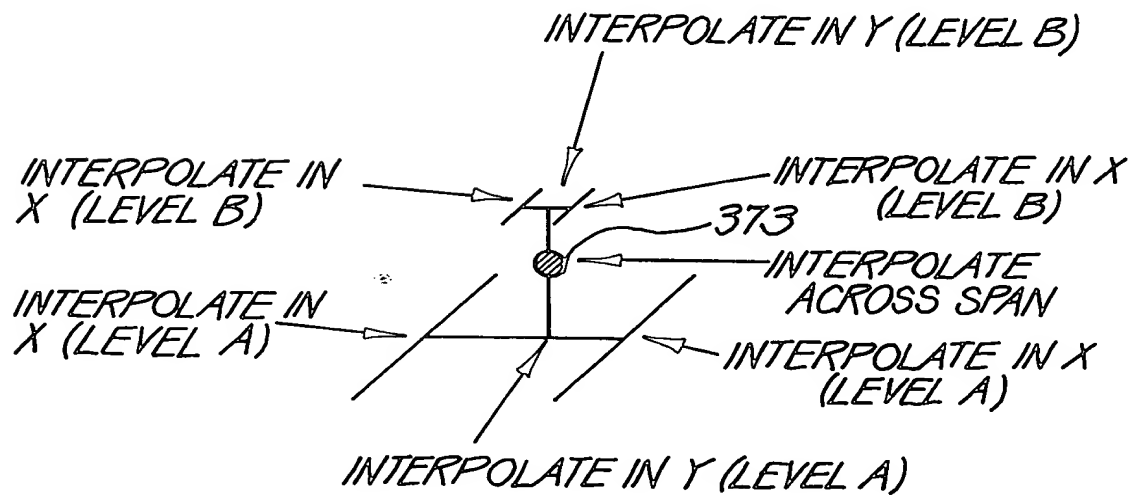


FIG. 56

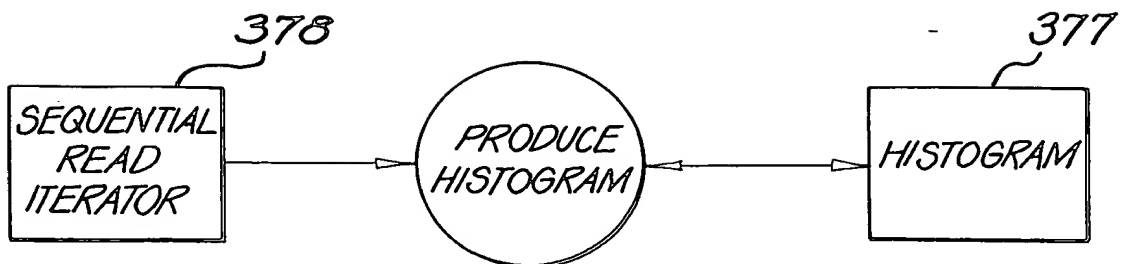


FIG. 57

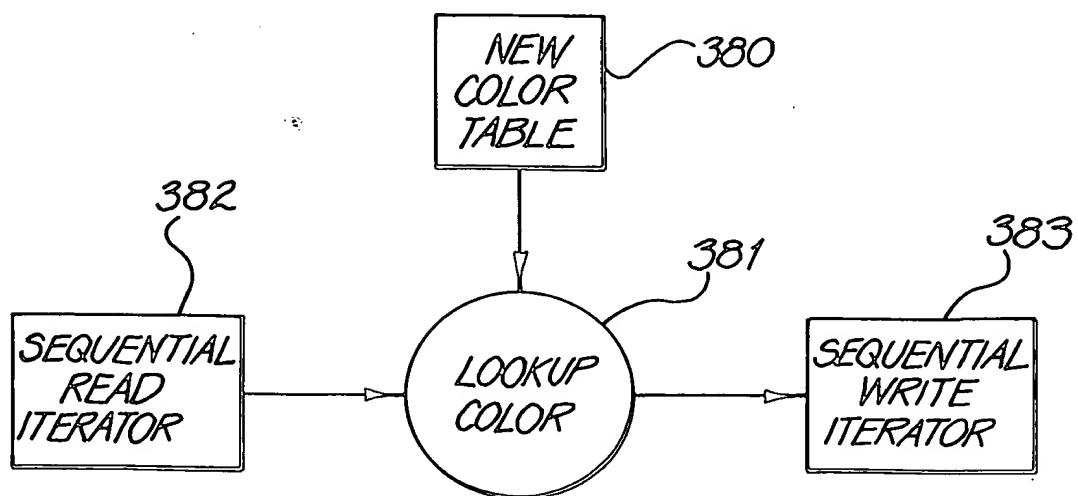


FIG. 58

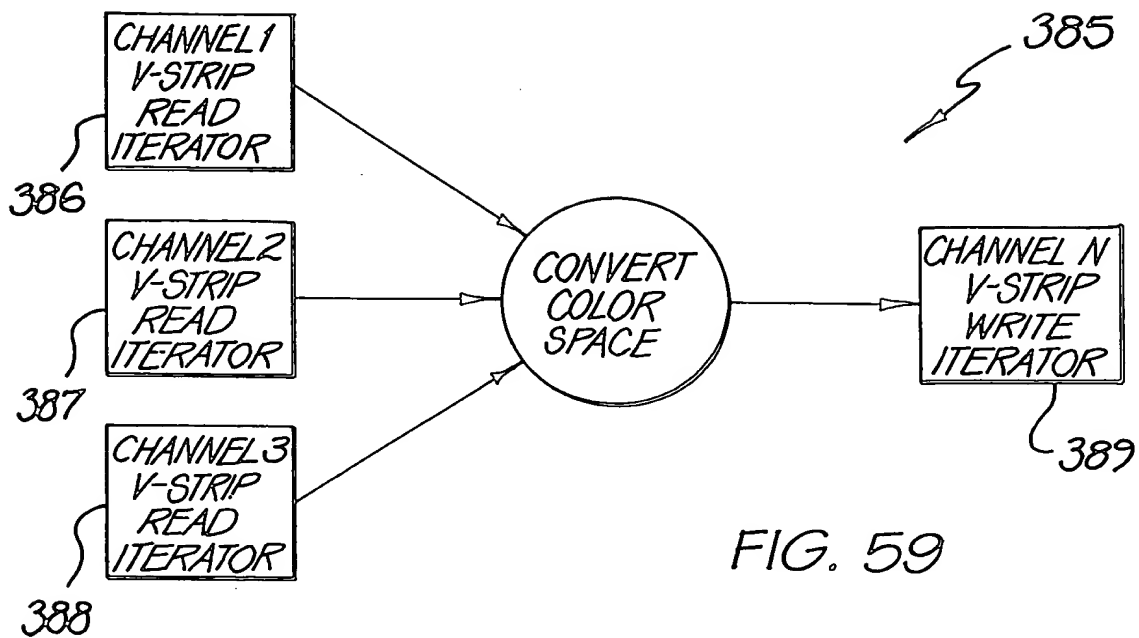


FIG. 59

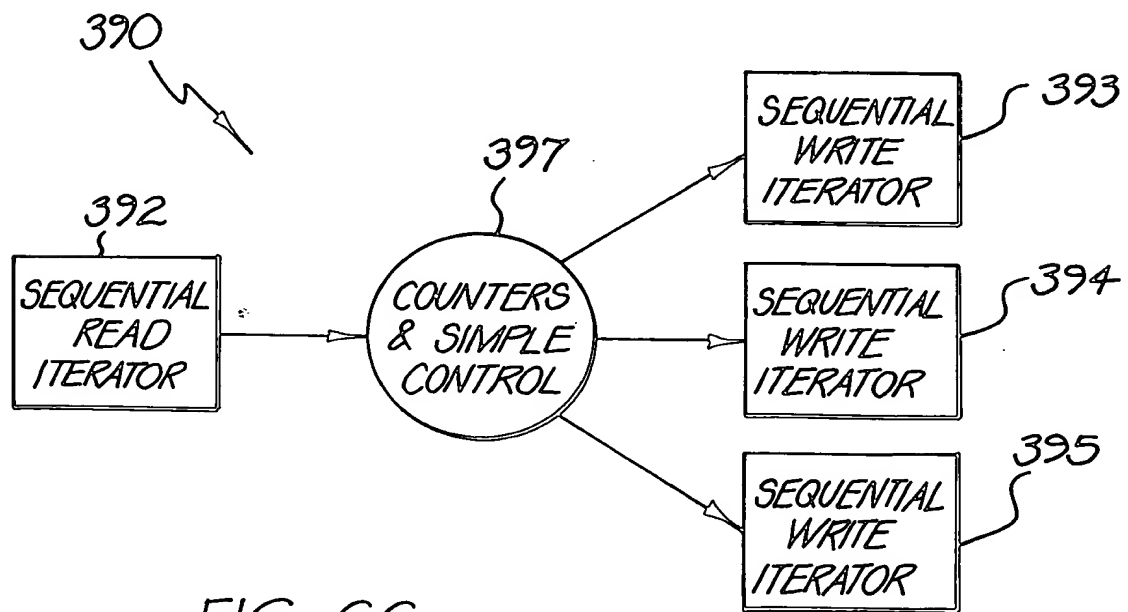


FIG. 66

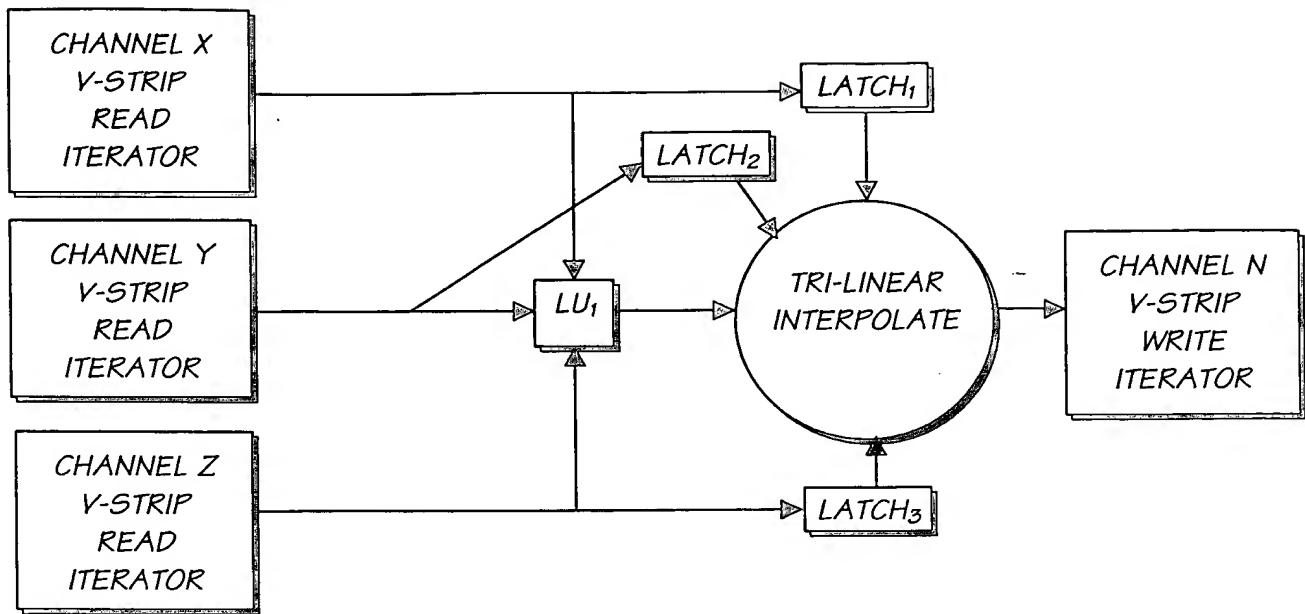


FIG. 60

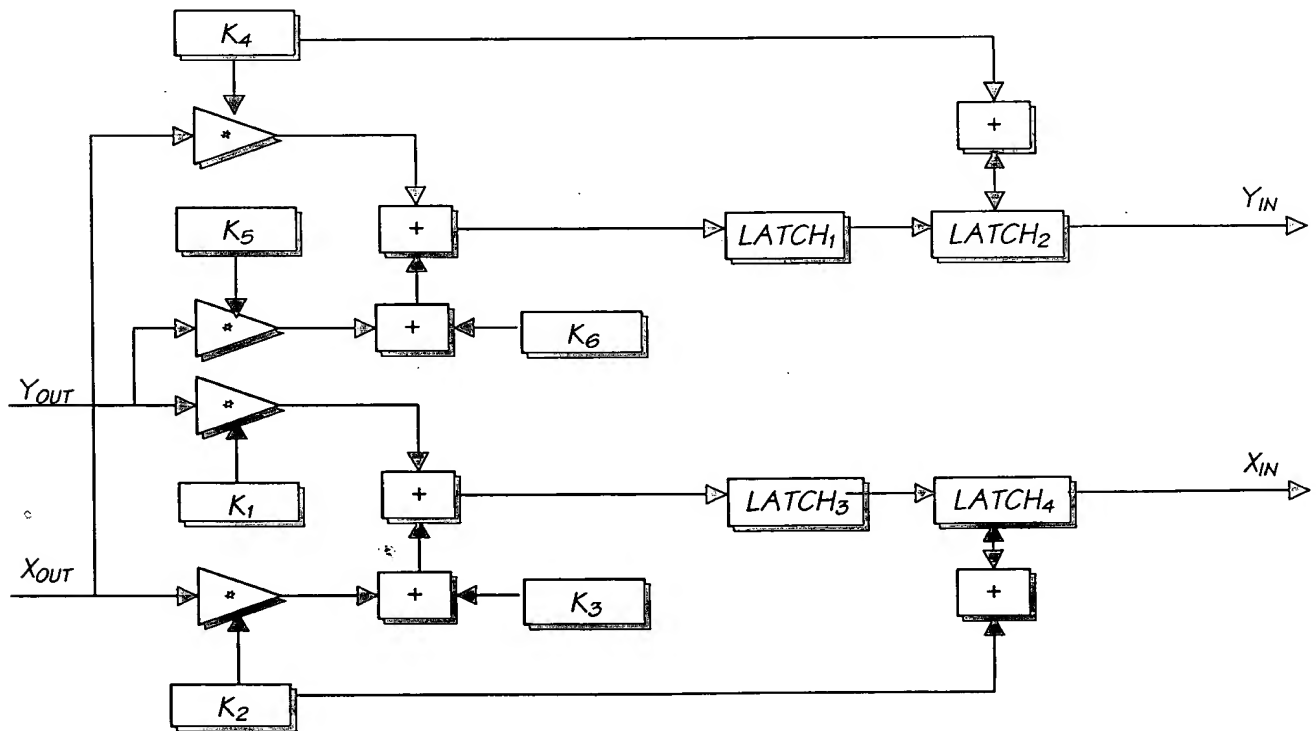


FIG. 61

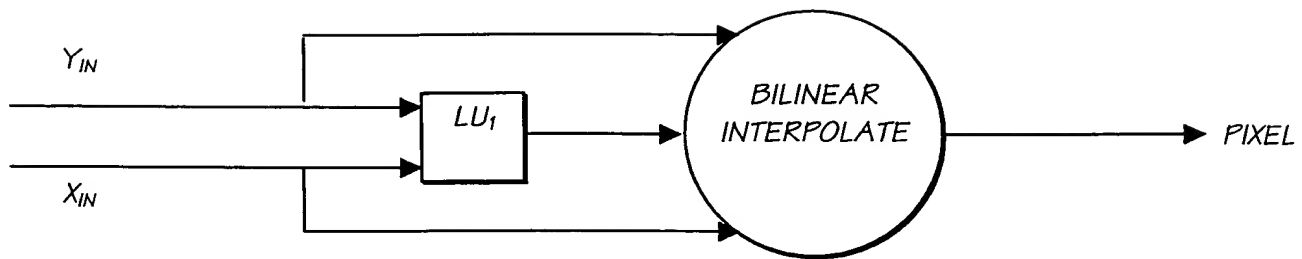


FIG. 62

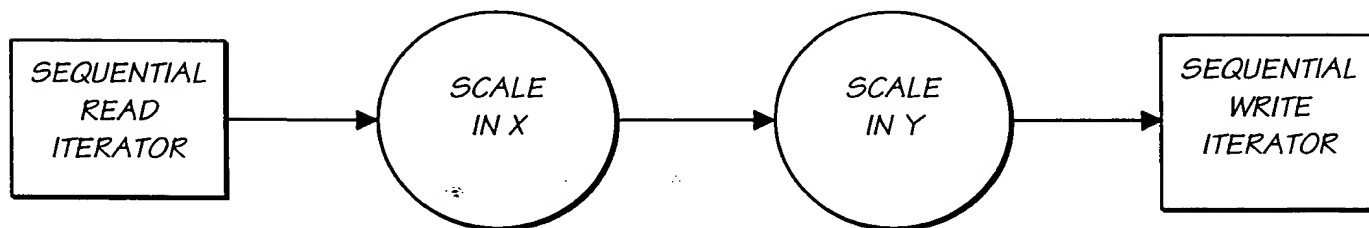


FIG. 63

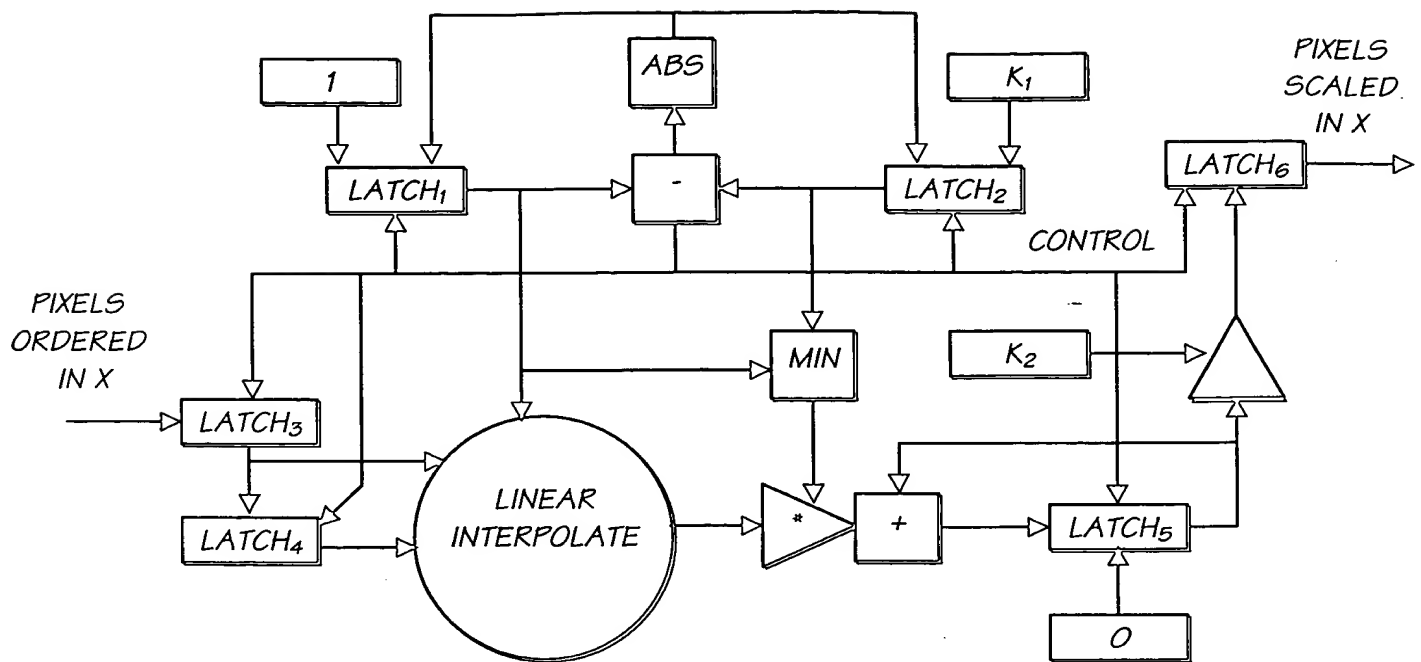


FIG. 64

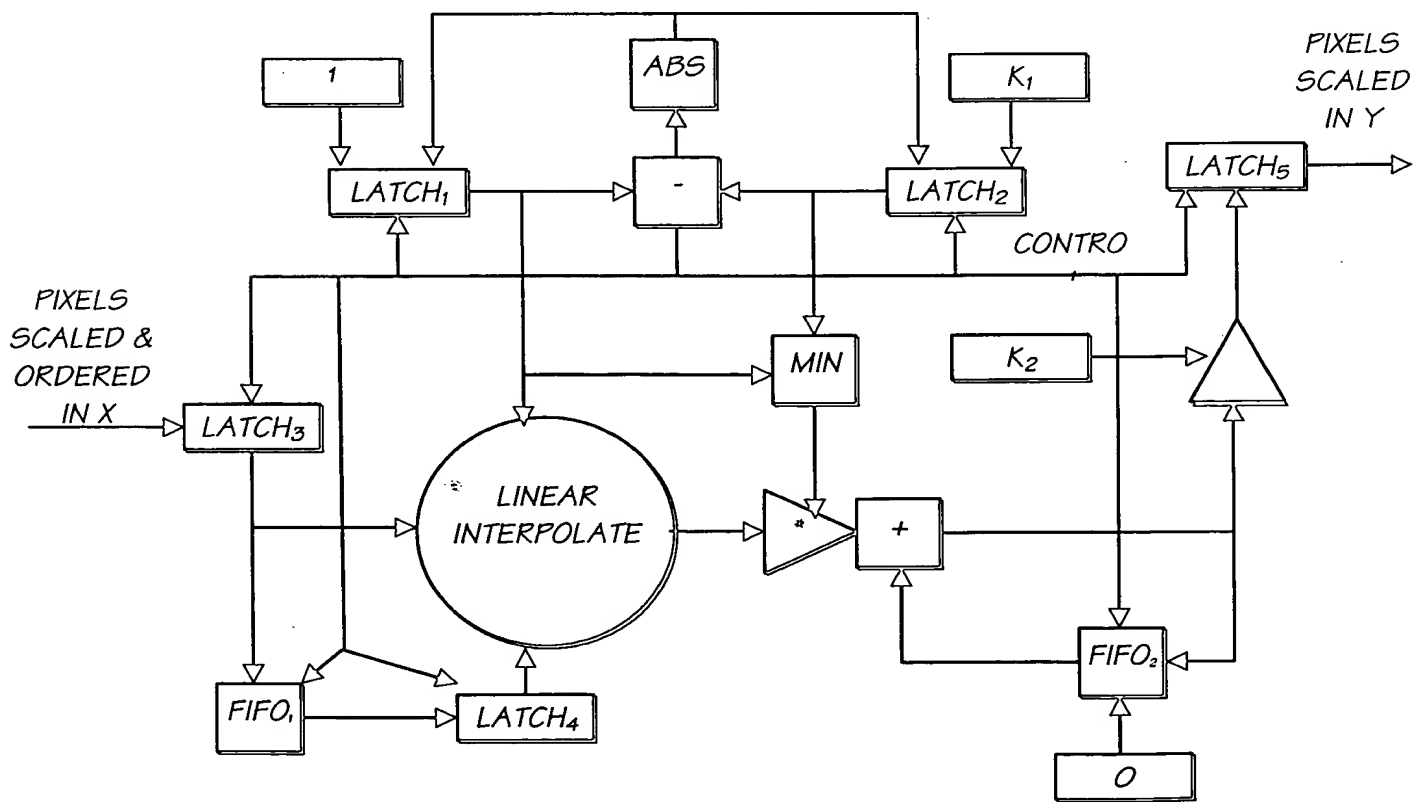
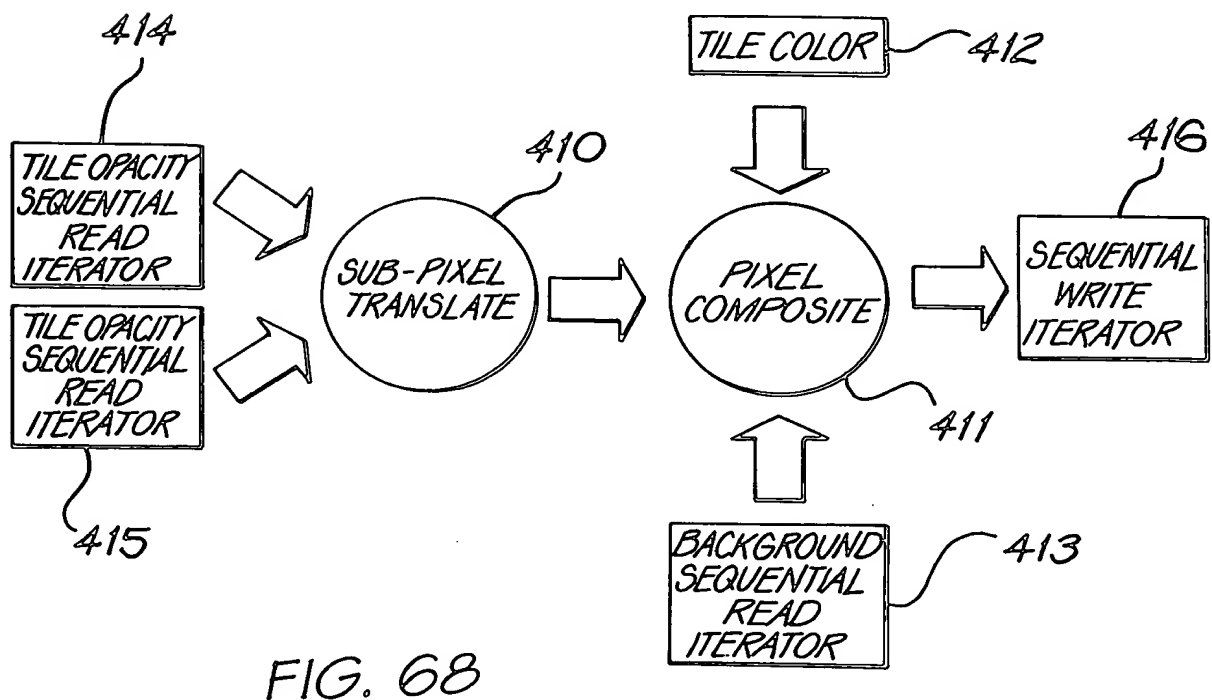
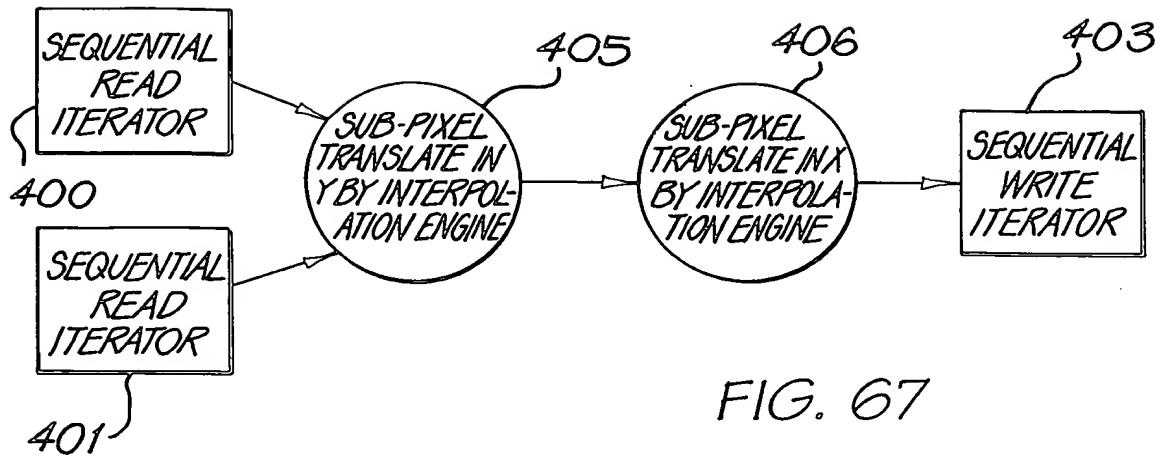


FIG. 65



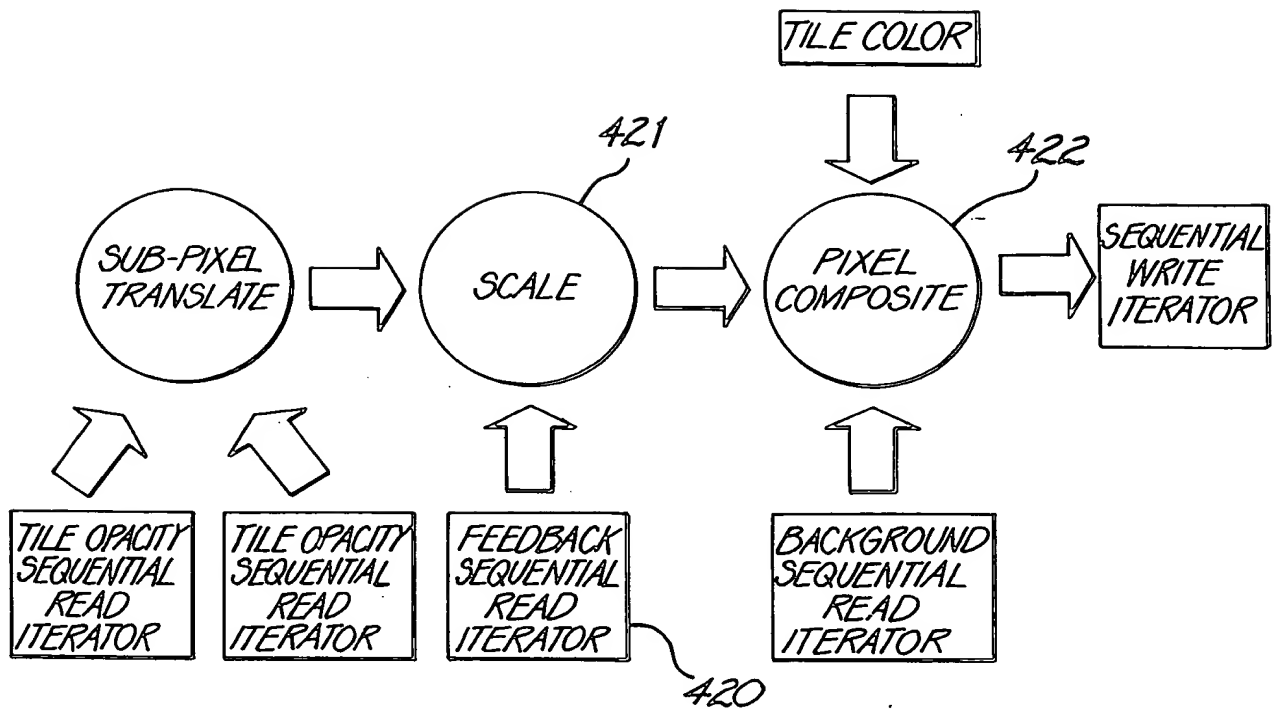


FIG. 69

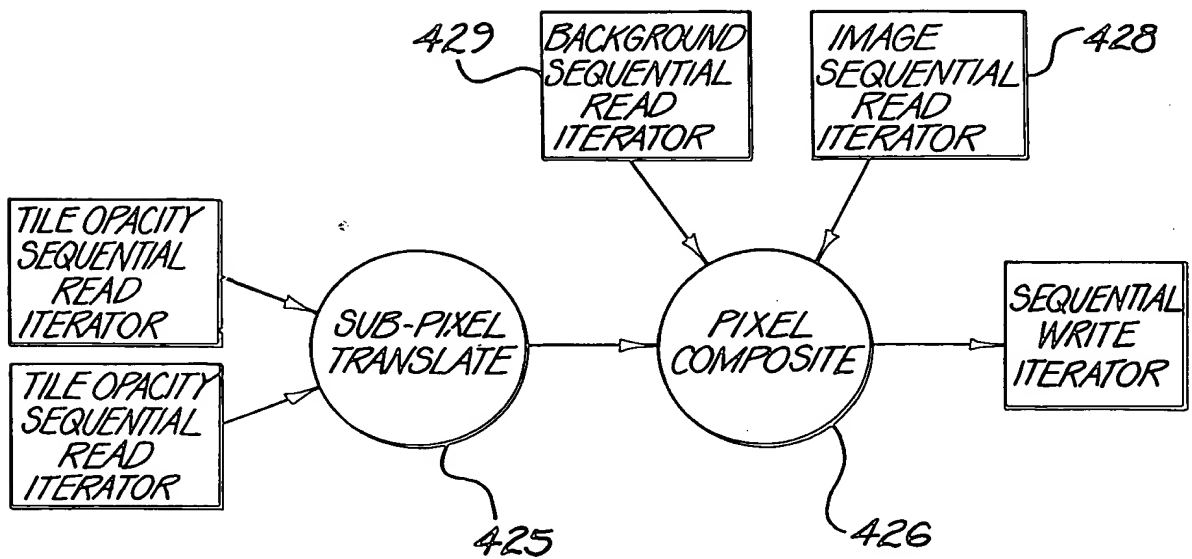


FIG. 70

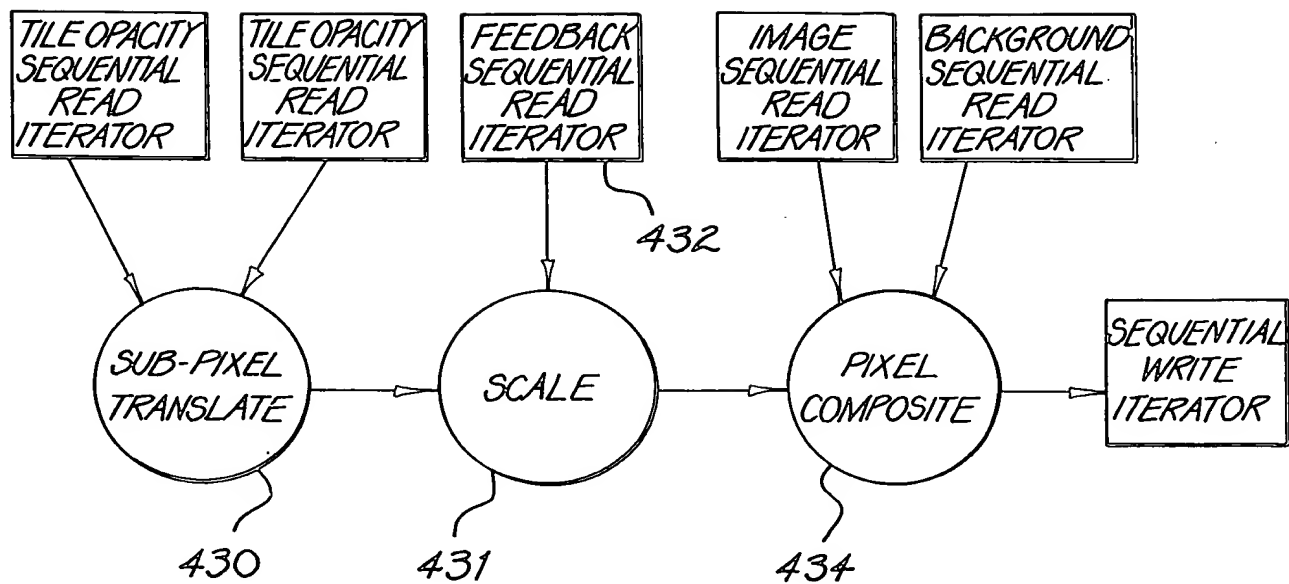


FIG. 71

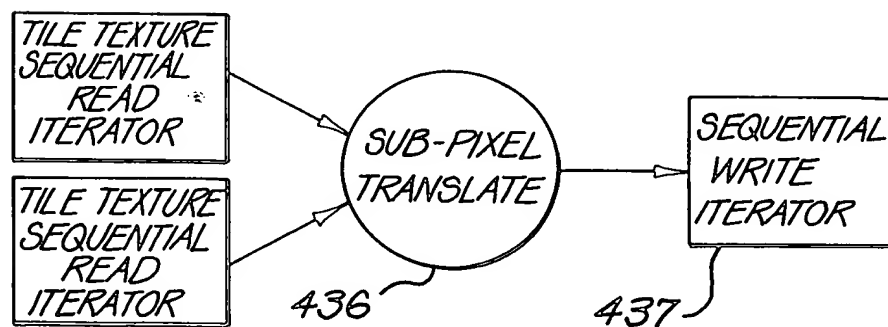


FIG. 72

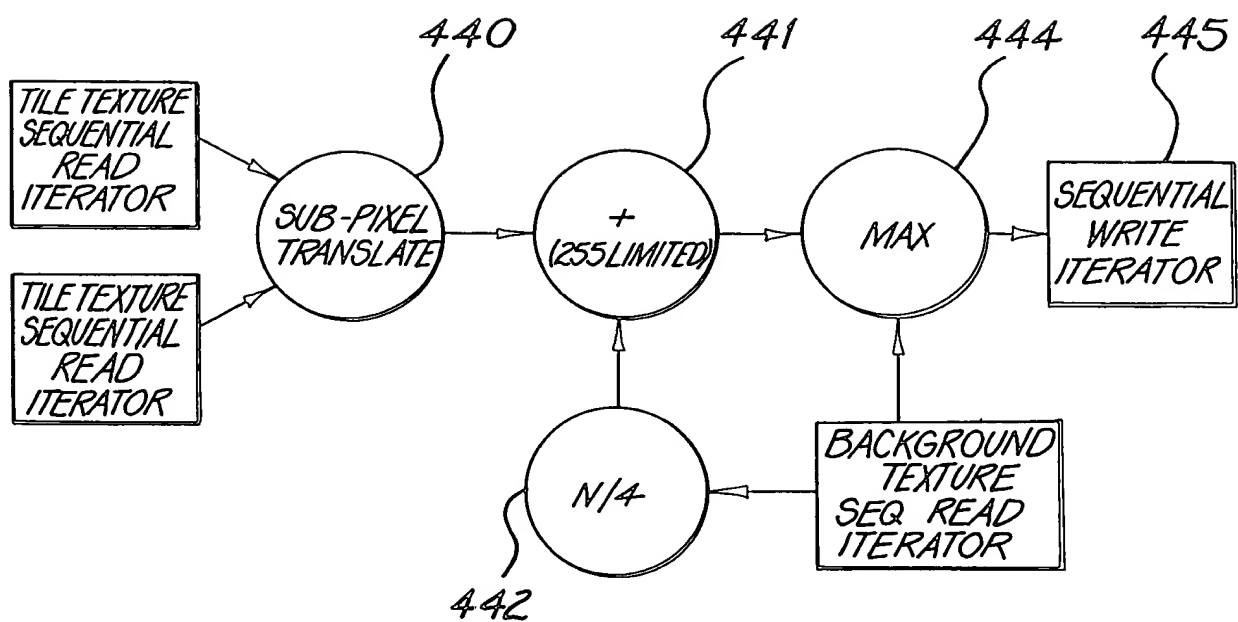


FIG. 73

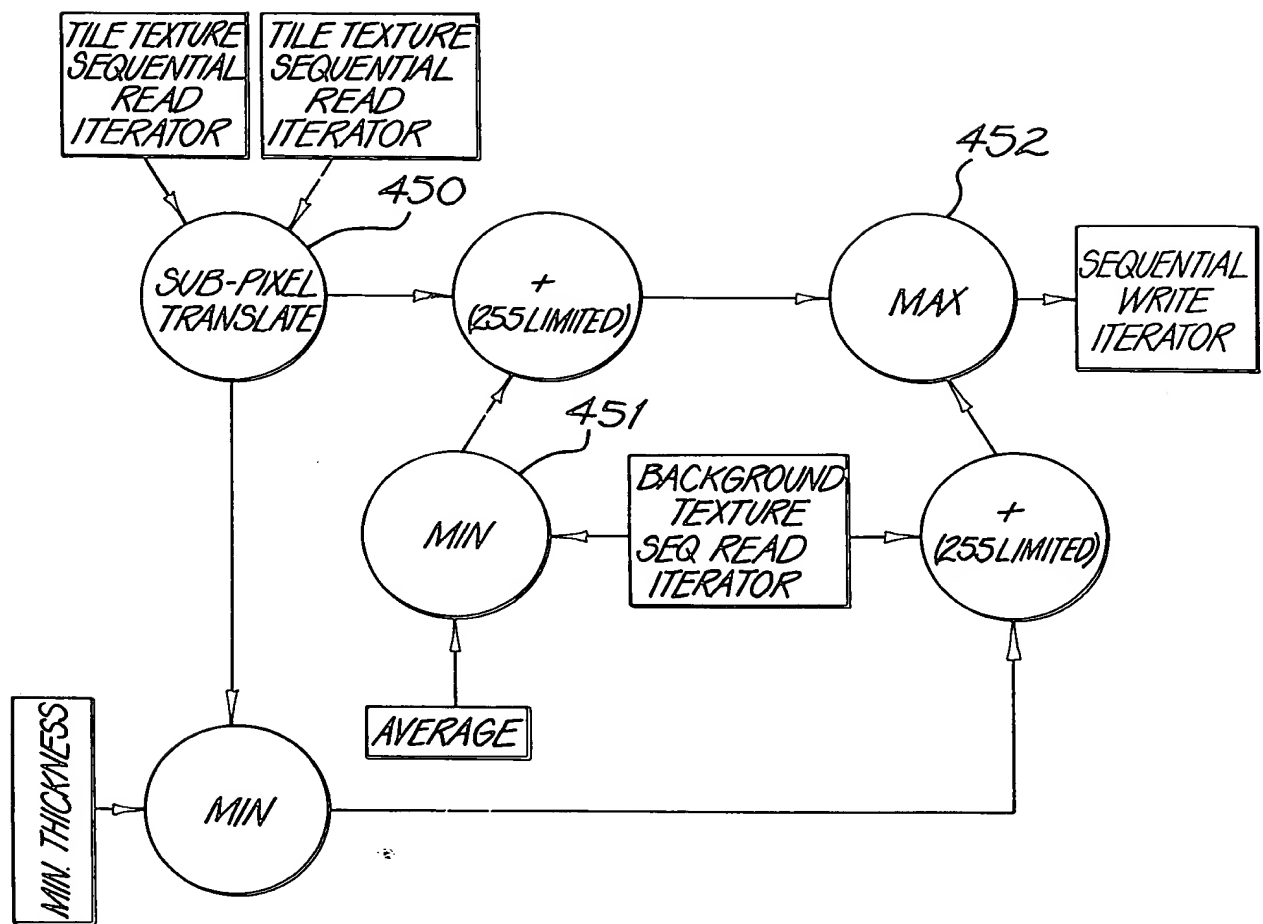


FIG. 74

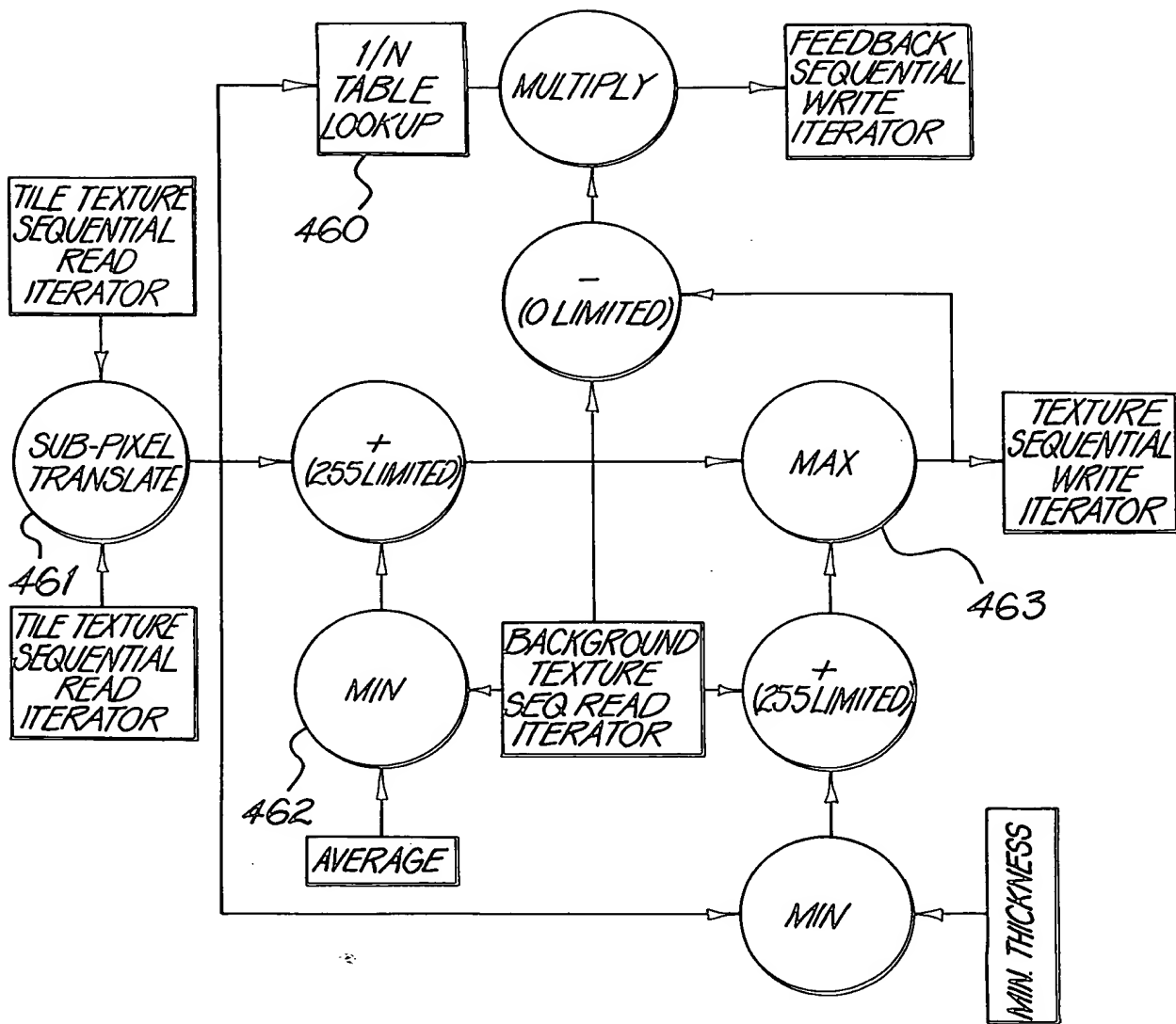


FIG. 75

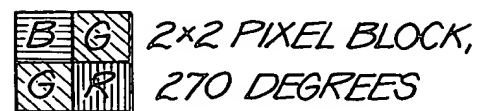
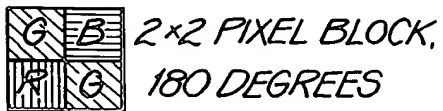
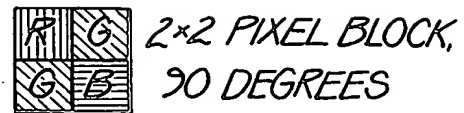
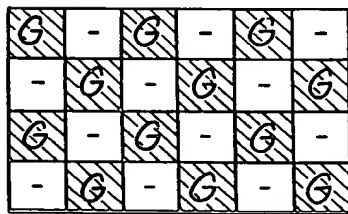


FIG. 76




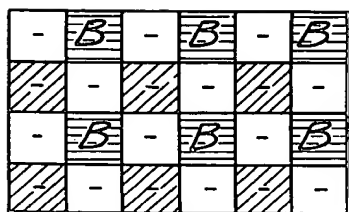
 LINEAR INTERPOLATED
PIXELS
 ACTUAL PIXELS (NOT
INTERPOLATED)

FIG. 77






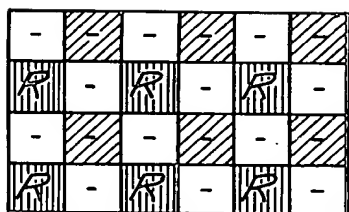
-  LINEAR INTERPOLATED PIXELS
-  BI-LINEAR INTERPOLATED PIXELS
-  ACTUAL PIXELS (NOT INTERPOLATED)

FIG. 78



-  LINEAR INTERPOLATED PIXELS
-  BI-LINEAR INTERPOLATED PIXELS
-  ACTUAL PIXELS (NOT INTERPOLATED)

FIG. 79

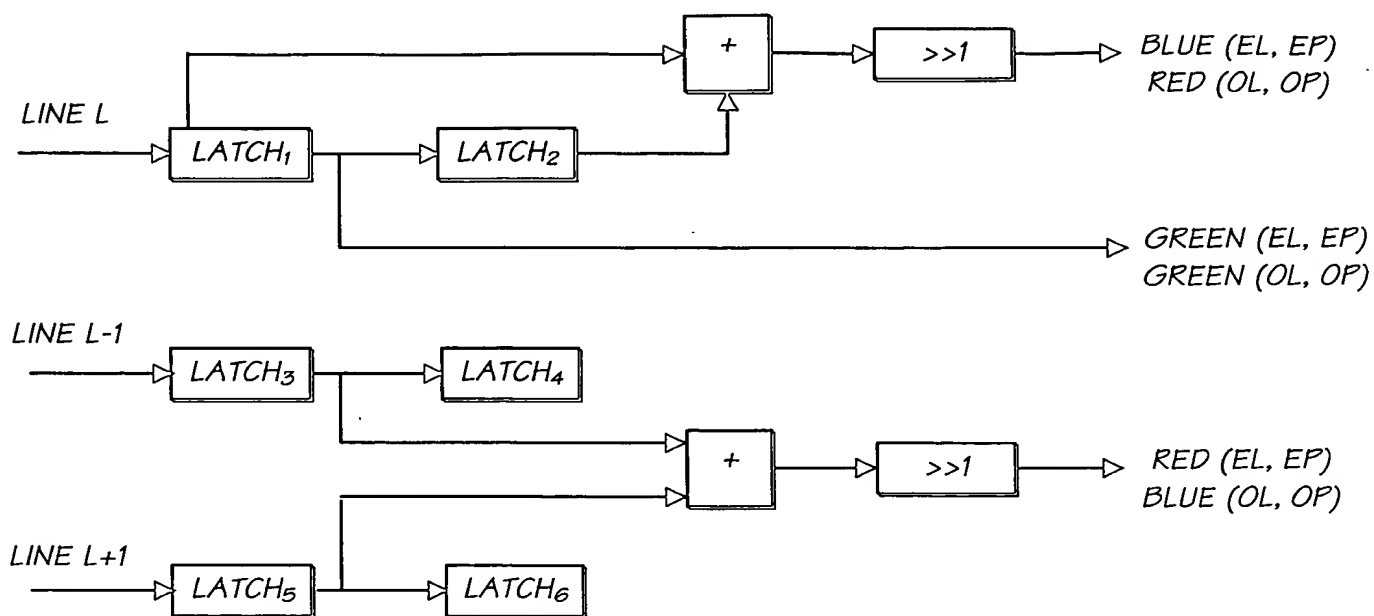


FIG. 80

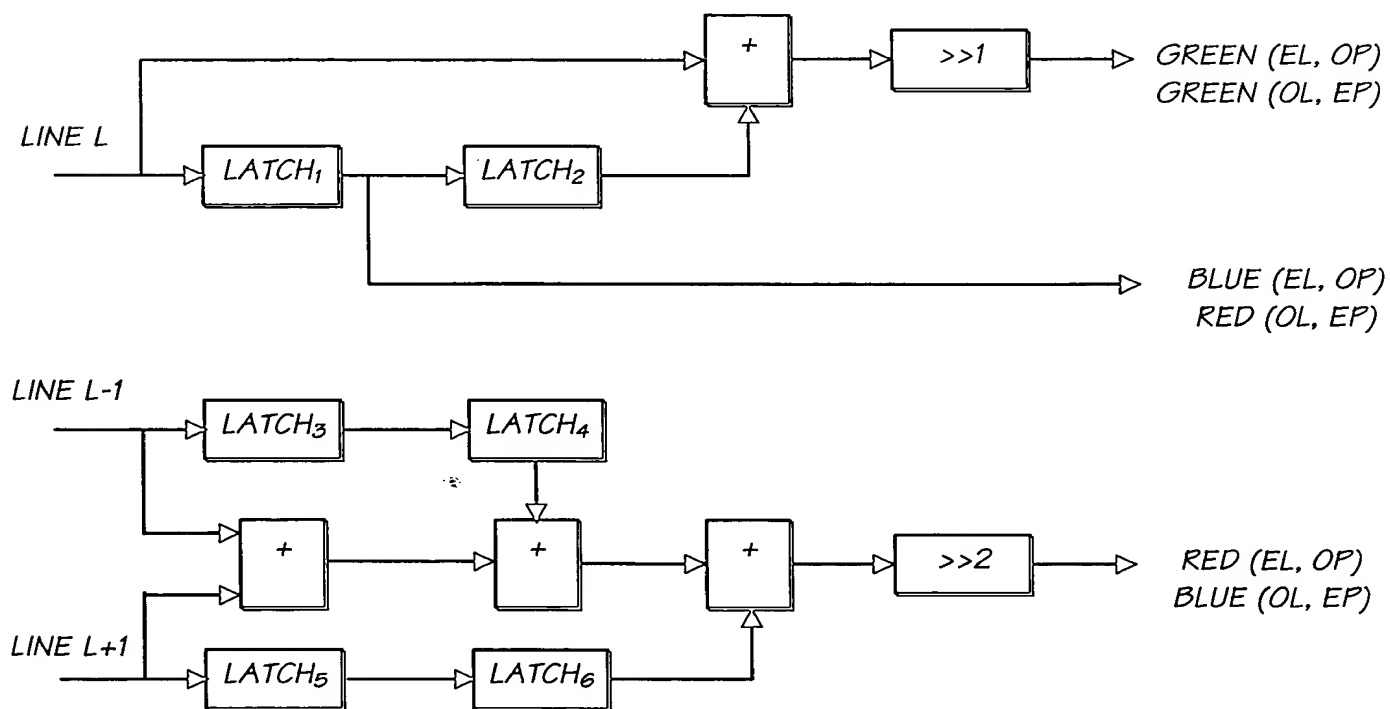


FIG. 81

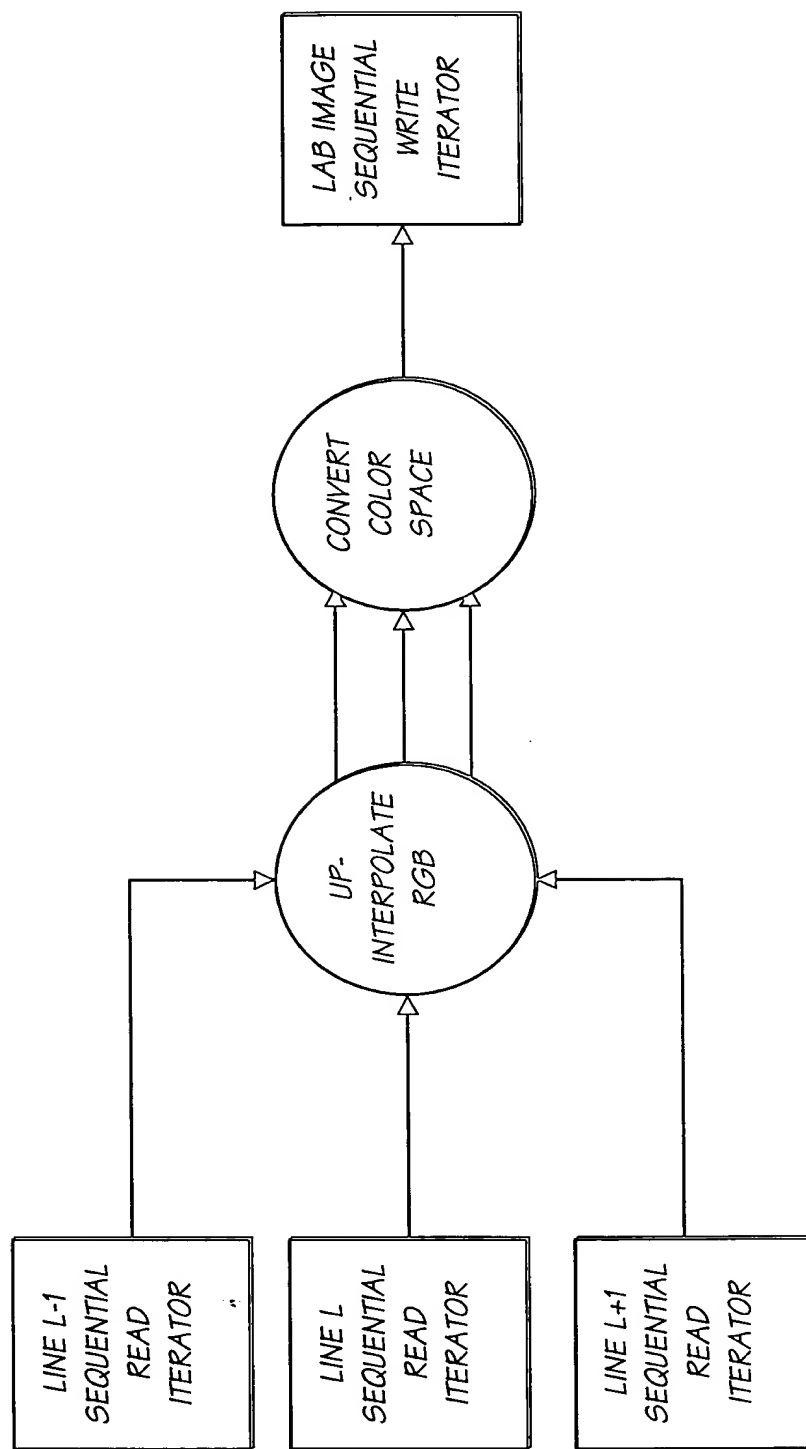


FIG. 82

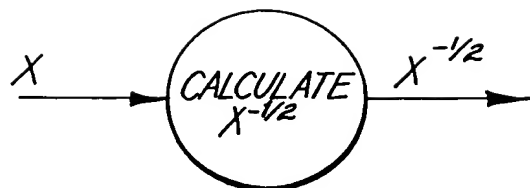


FIG. 83

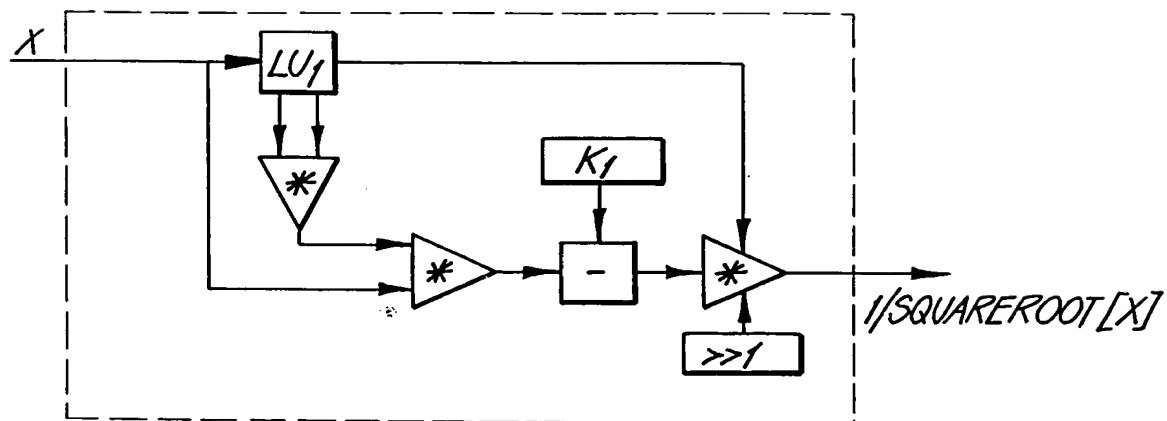


FIG. 84

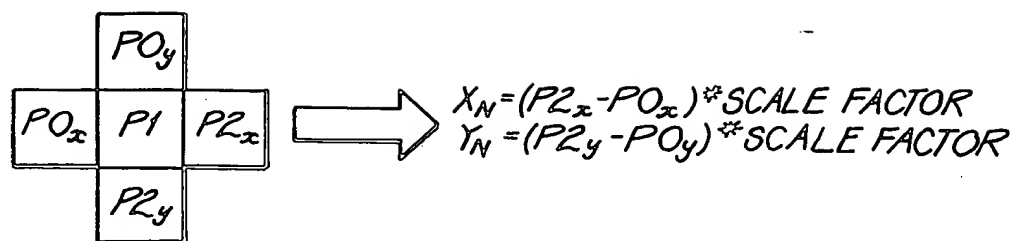


FIG. 85

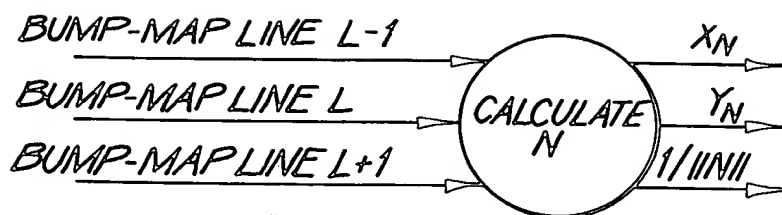


FIG. 86

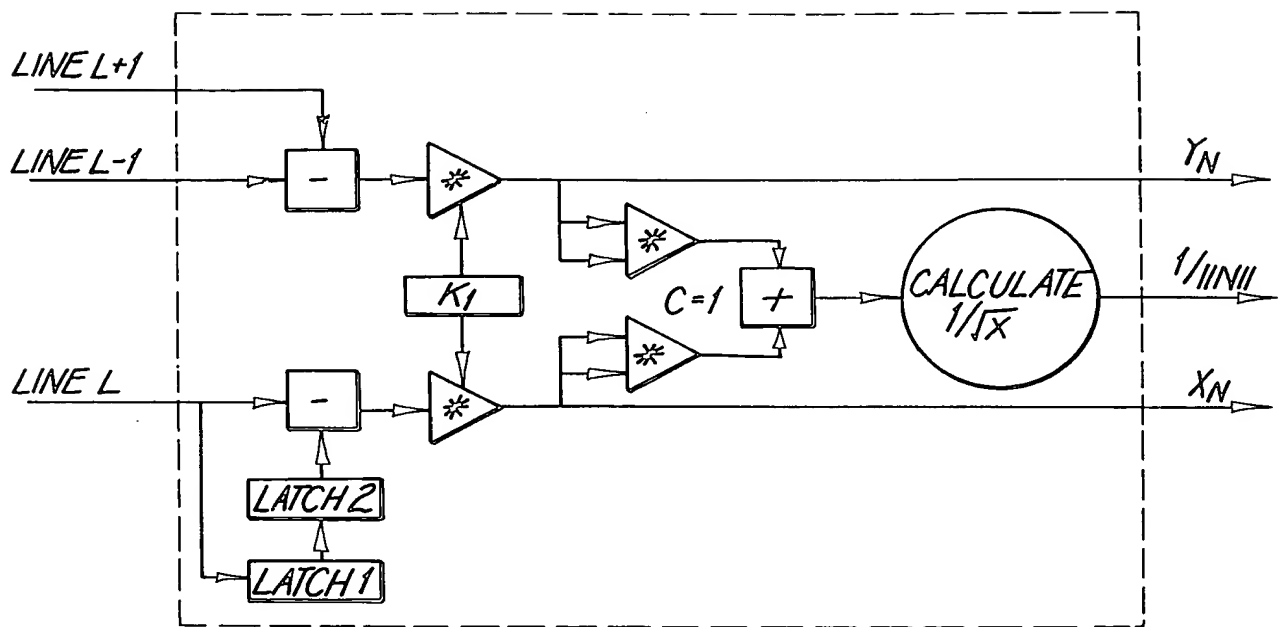


FIG. 87

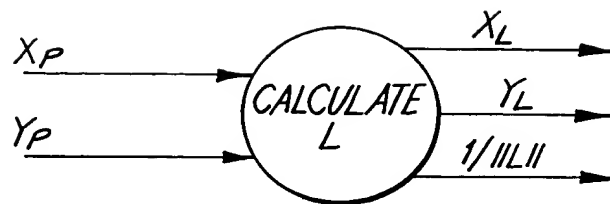


FIG. 88

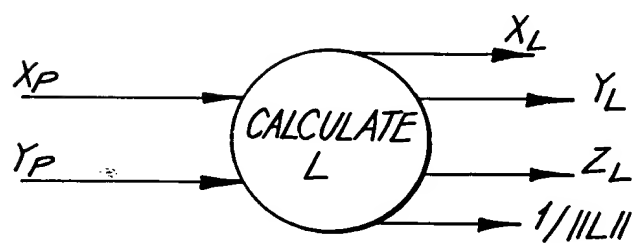


FIG. 89

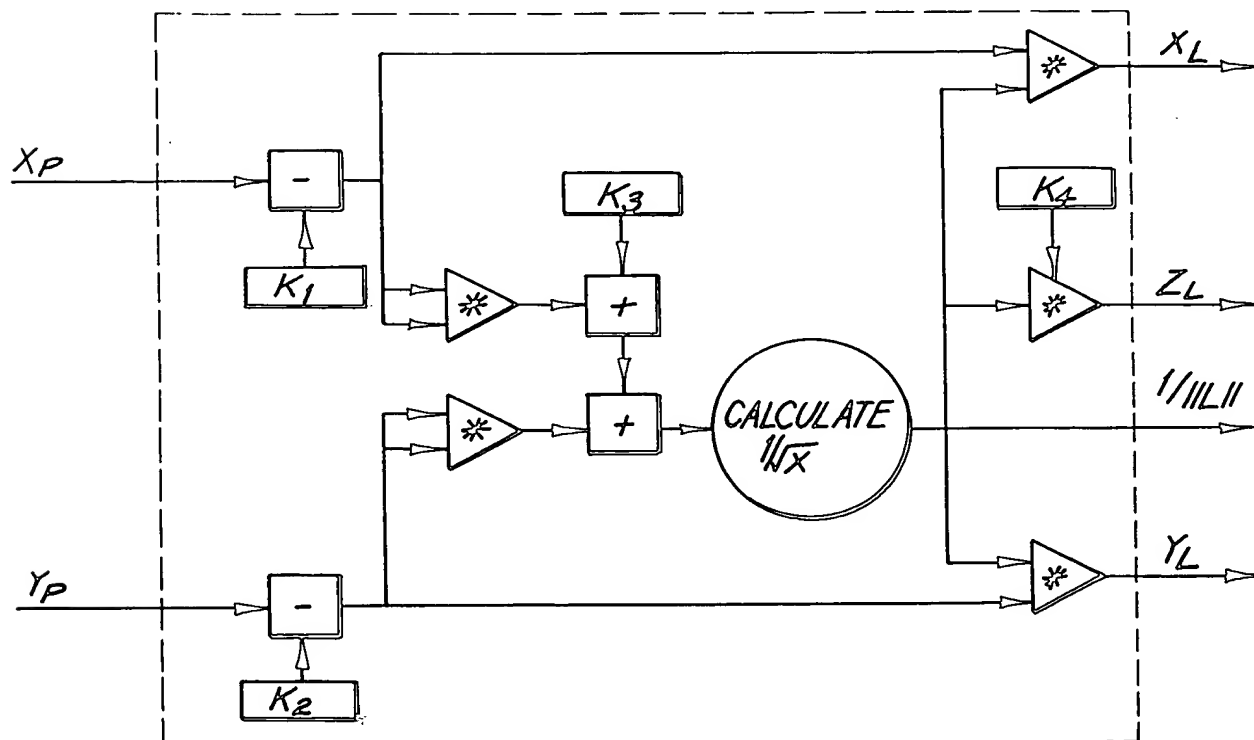


FIG. 90

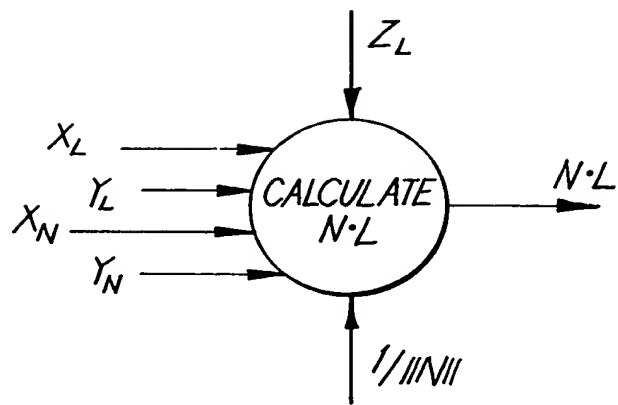


FIG. 91

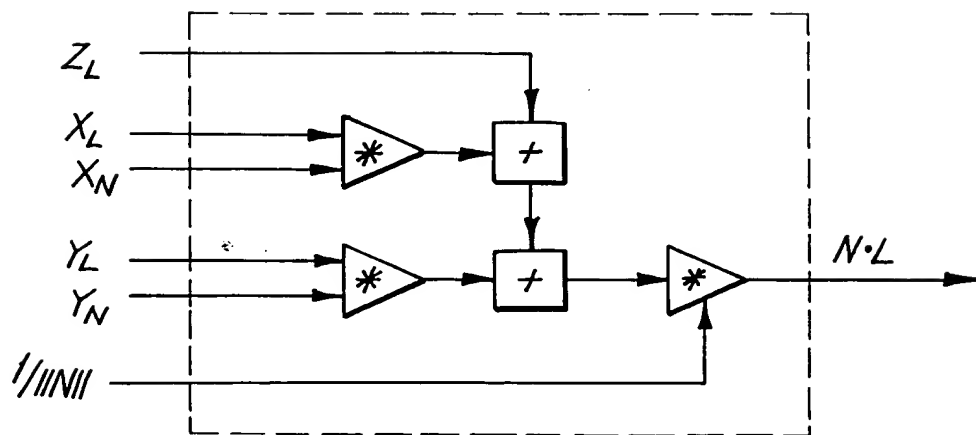


FIG. 92

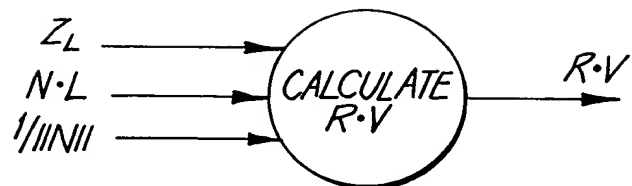


FIG. 93

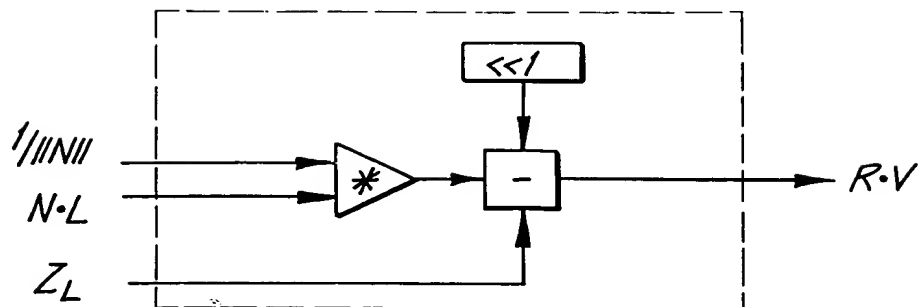


FIG. 94

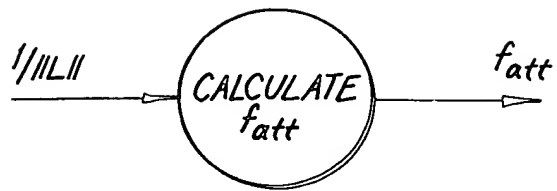


FIG. 95

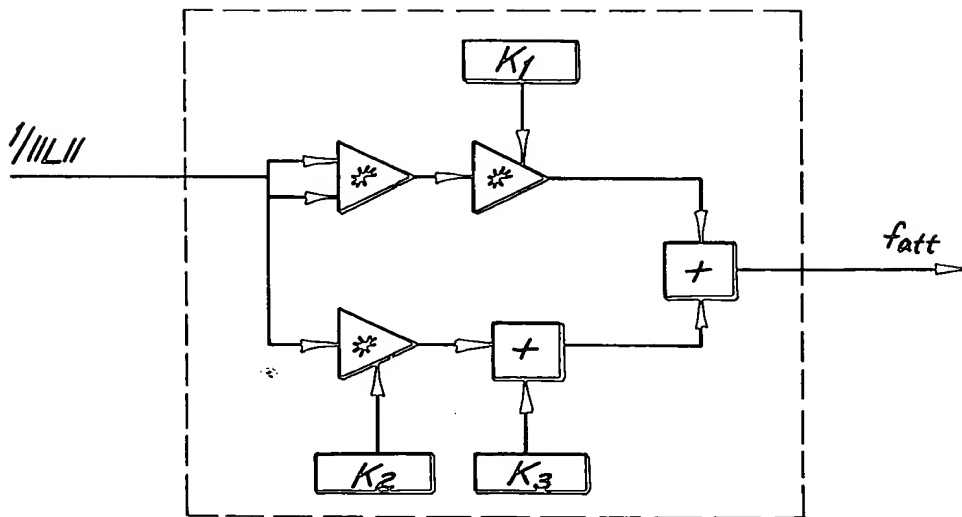
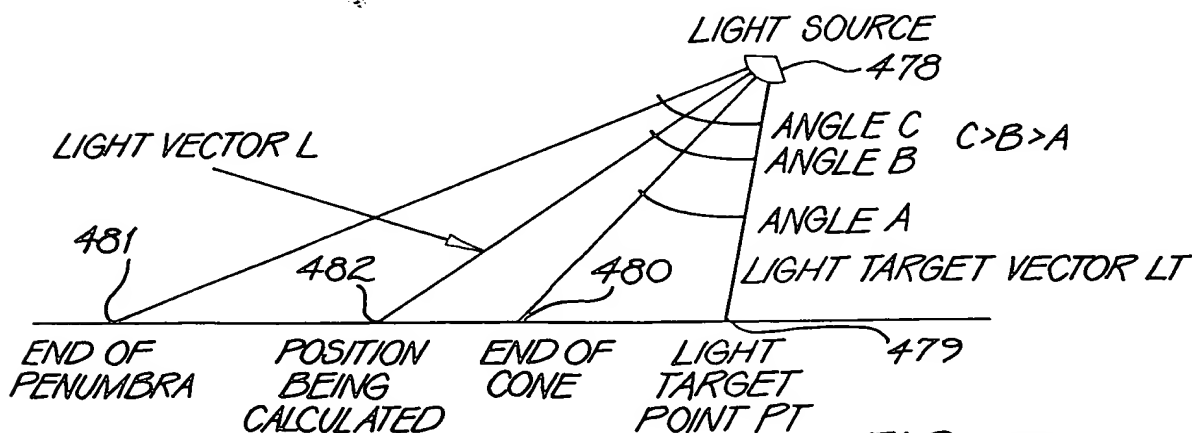
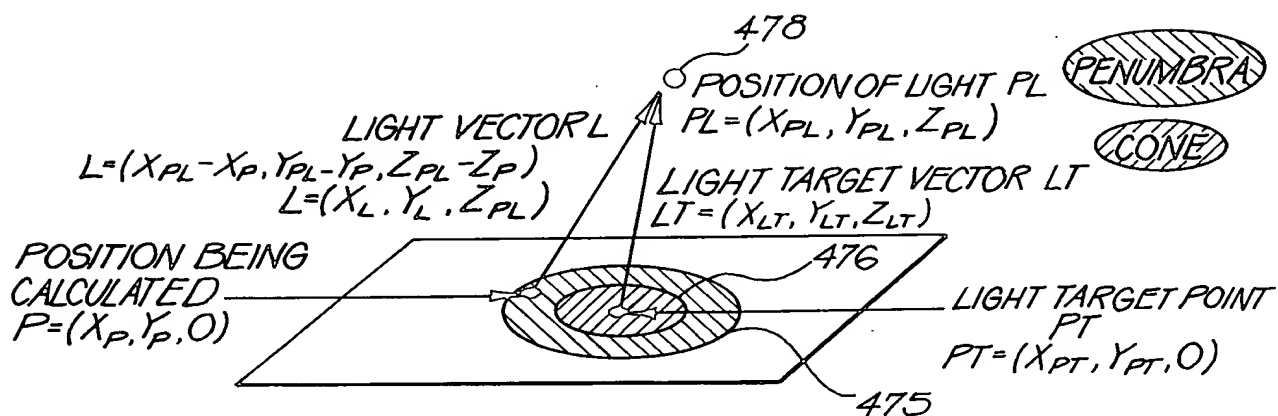
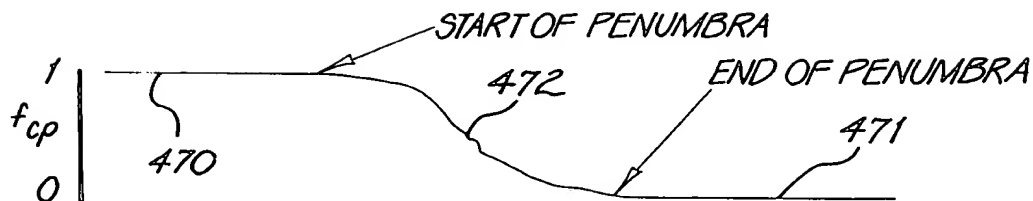


FIG. 96



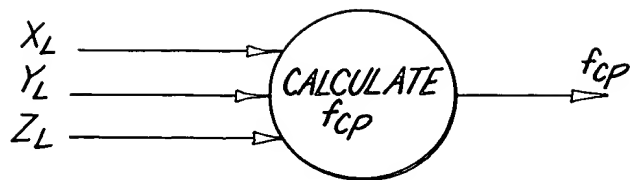


FIG. 100

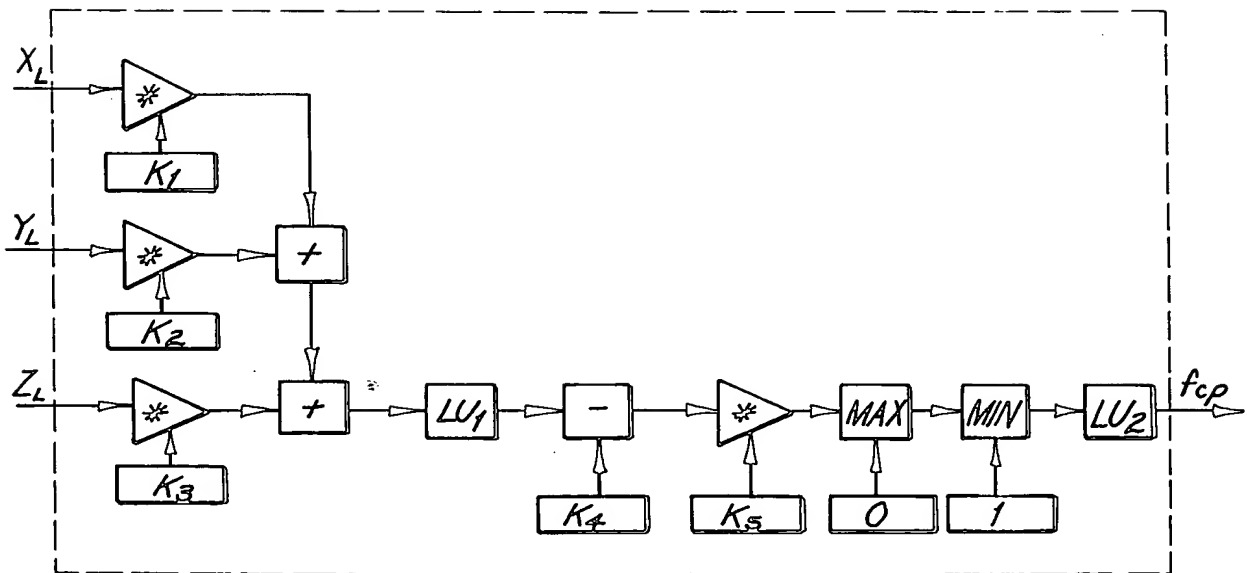


FIG. 101

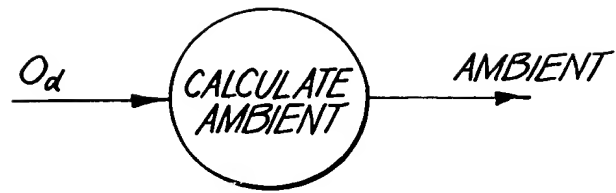


FIG. 102

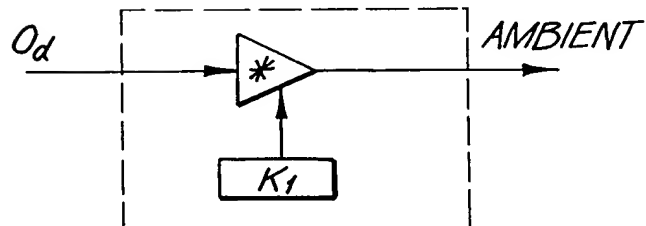


FIG. 103

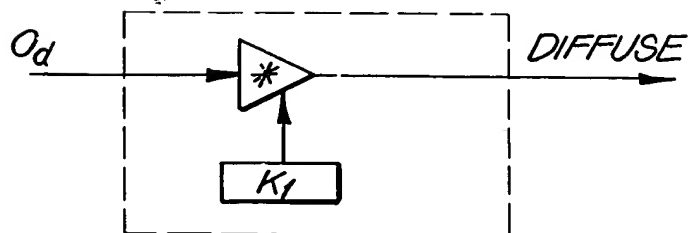


FIG. 104

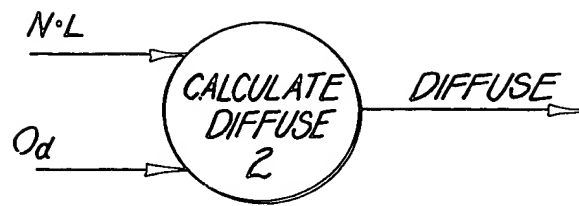


FIG. 105

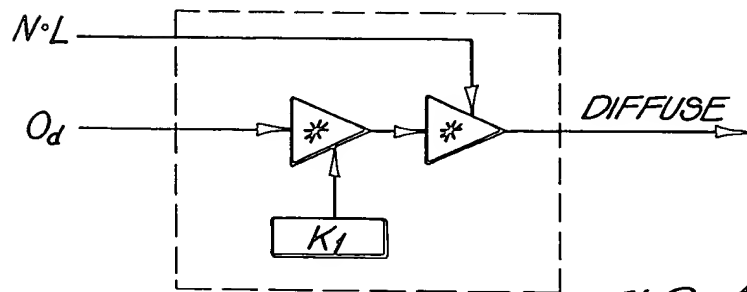


FIG. 106

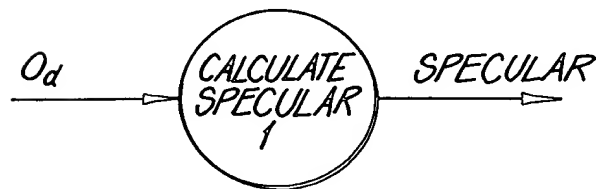


FIG. 107

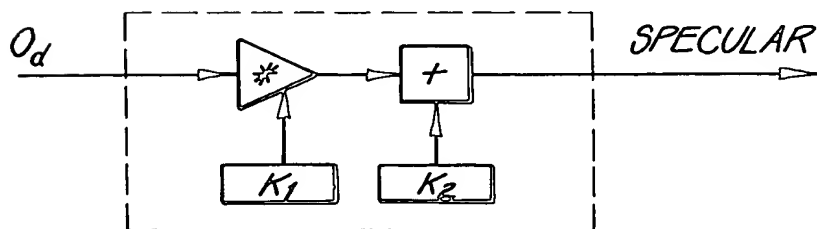


FIG. 108

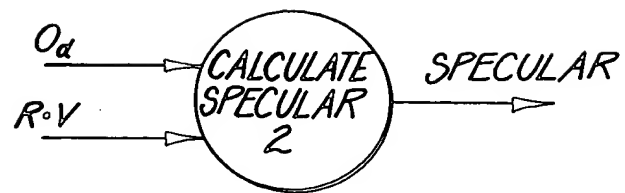


FIG. 109

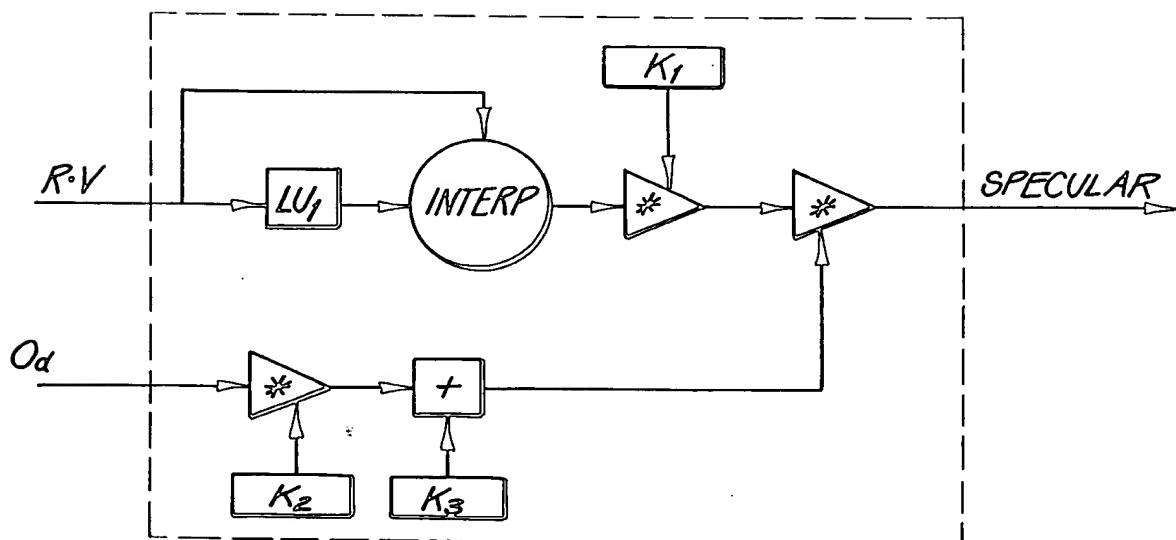


FIG. 110

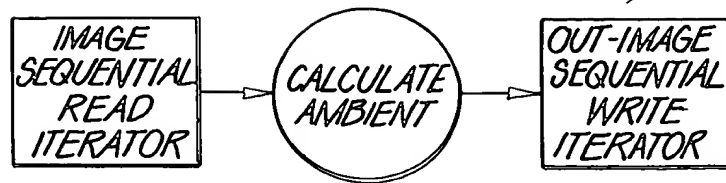


FIG. 111

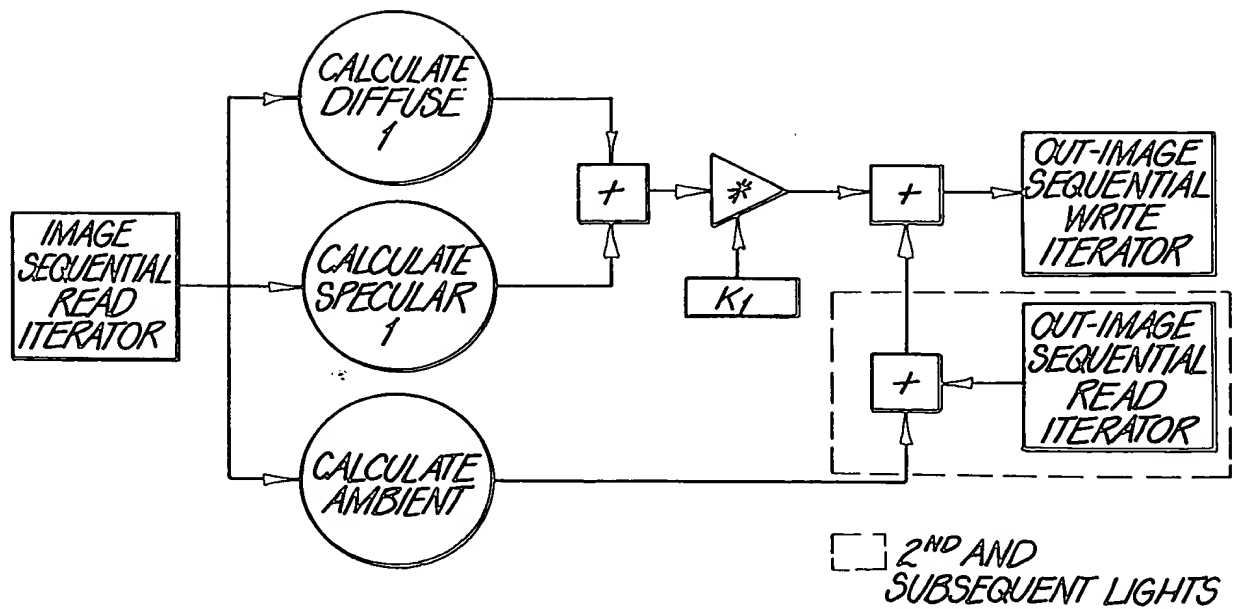


FIG. 112

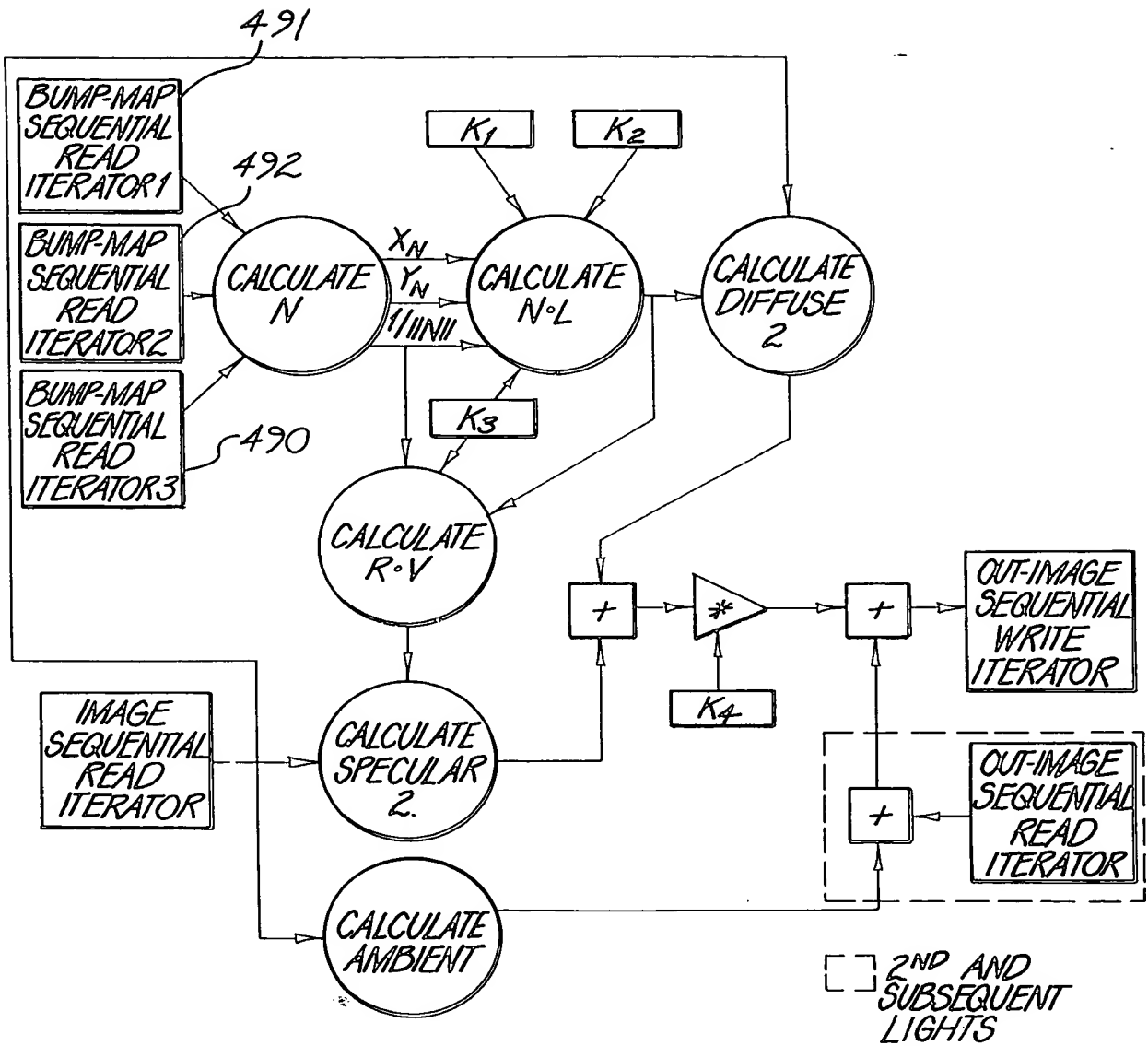


FIG. 113

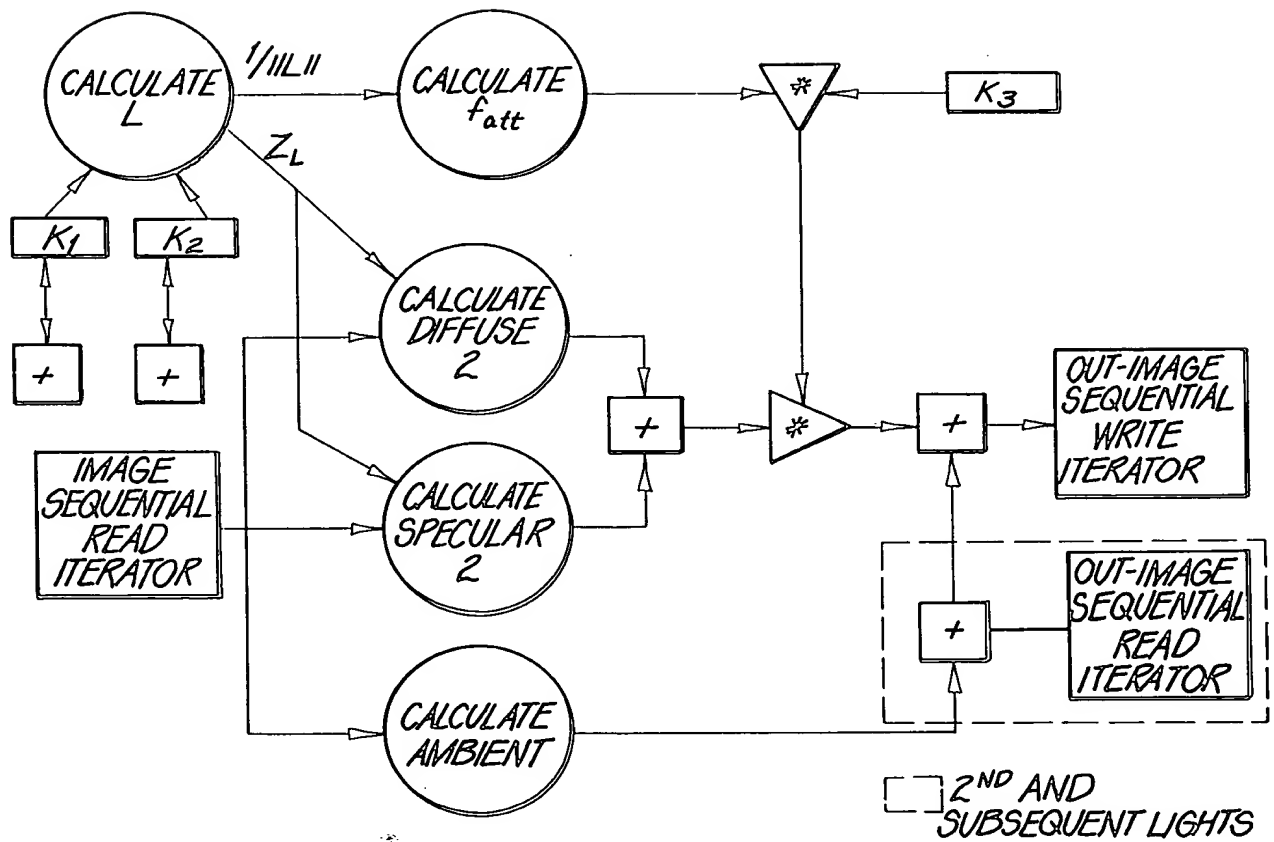


FIG. 114

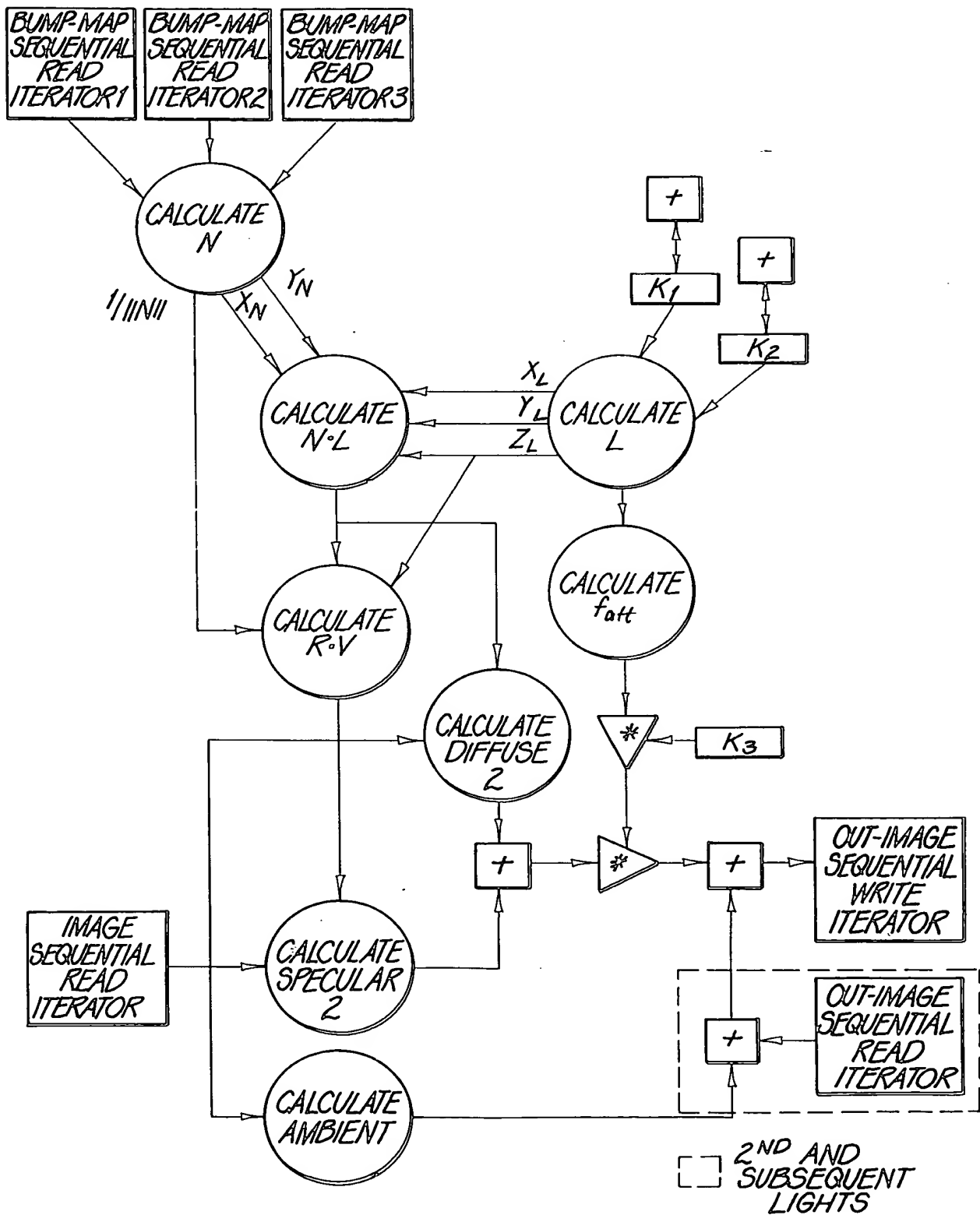


FIG. 115

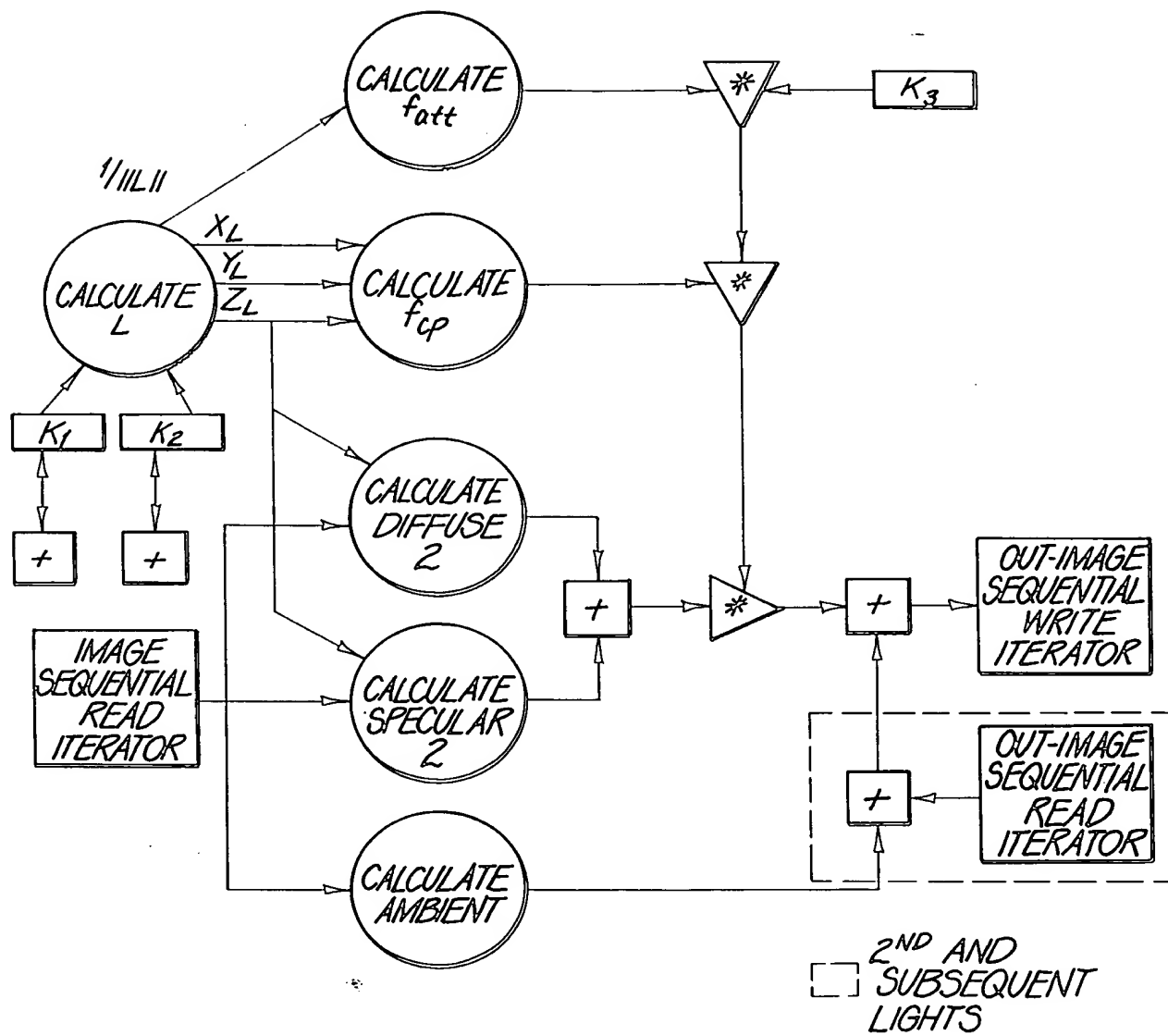
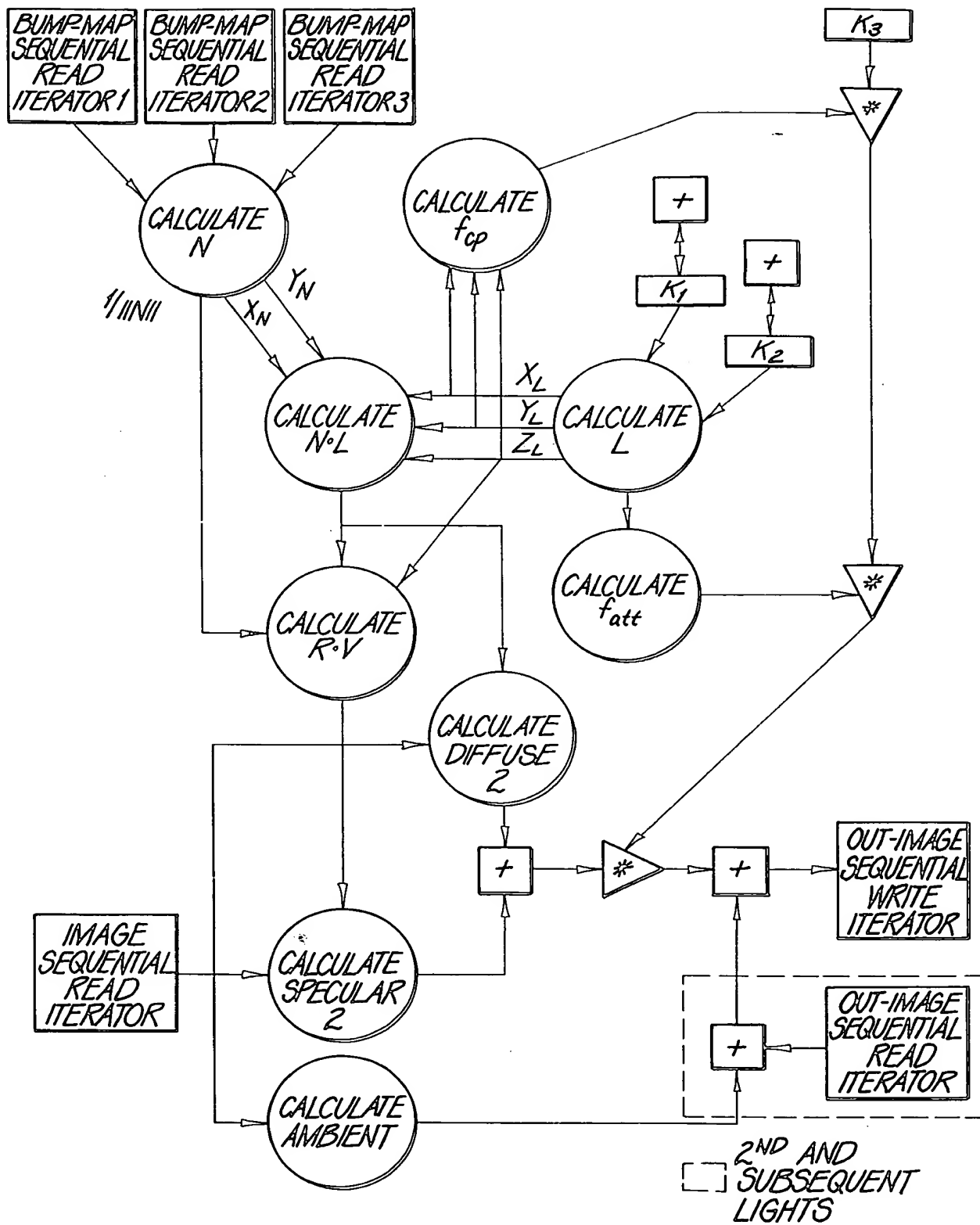
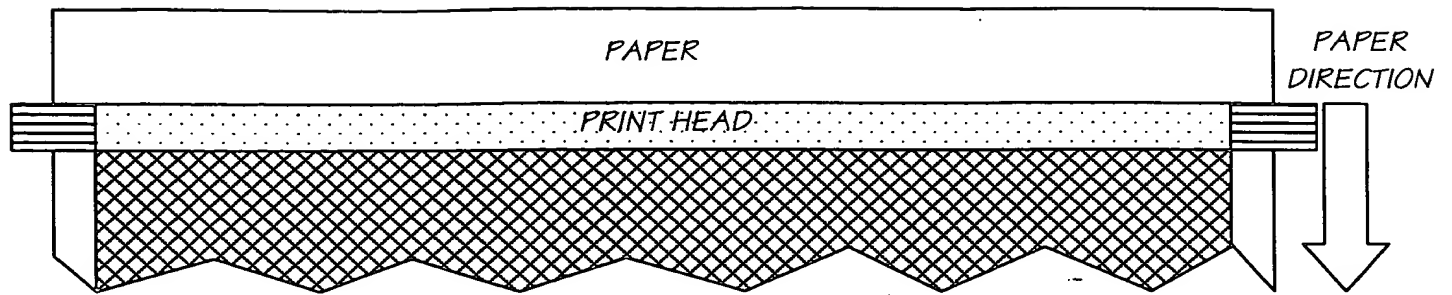


FIG. 116



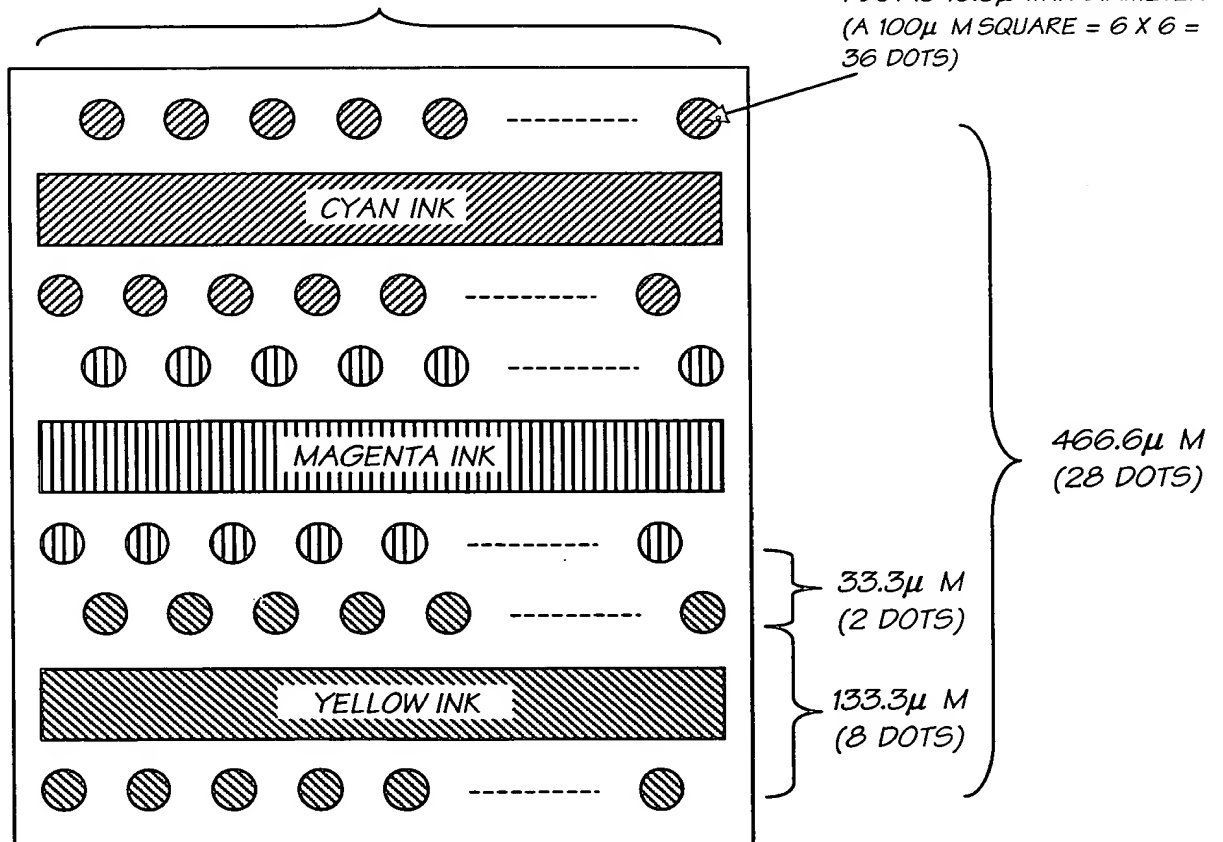


8 PRINT HEAD SEGMENTS IN PRINT HEAD

SEGMENT 0	SEGMENT 1	SEGMENT 2	SEGMENT 3	SEGMENT 4	SEGMENT 5	SEGMENT 6	SEGMENT 7
-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------

1250 μ M
(375 DOTS PER SEGMENT ROW, OR 750 DOTS PER SEGMENT)

1 DOT IS 16.6 μ M IN DIAMETER
(A 100 μ M SQUARE = 6 X 6 = 36 DOTS)



EACH SEGMENT CONTAINS 6 ROWS OF DOTS: ODD AND EVEN CYAN, MAGENTA, AND YELLOW.

FIG. 118

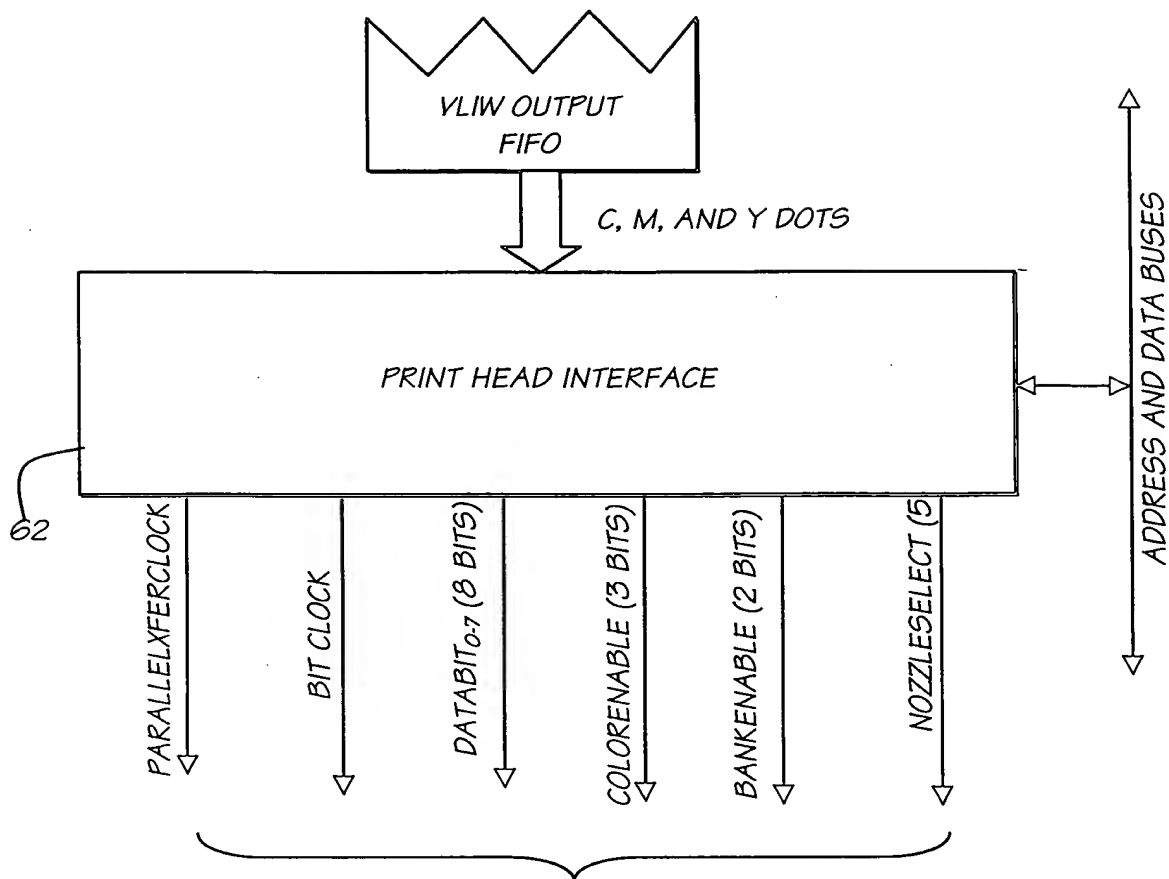


FIG. 119

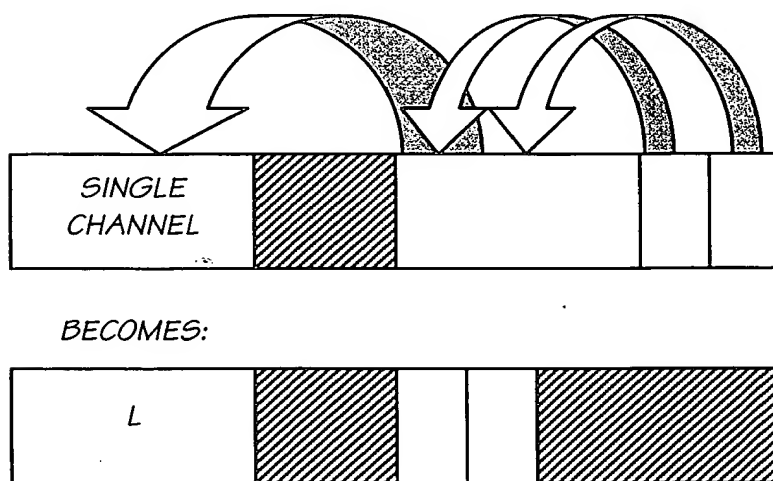


FIG. 120

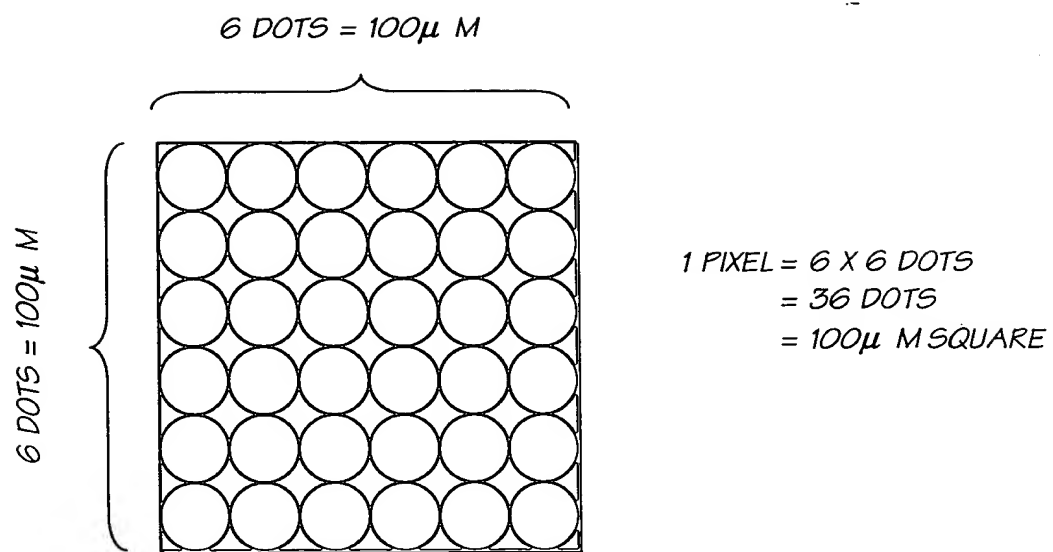


FIG. 121

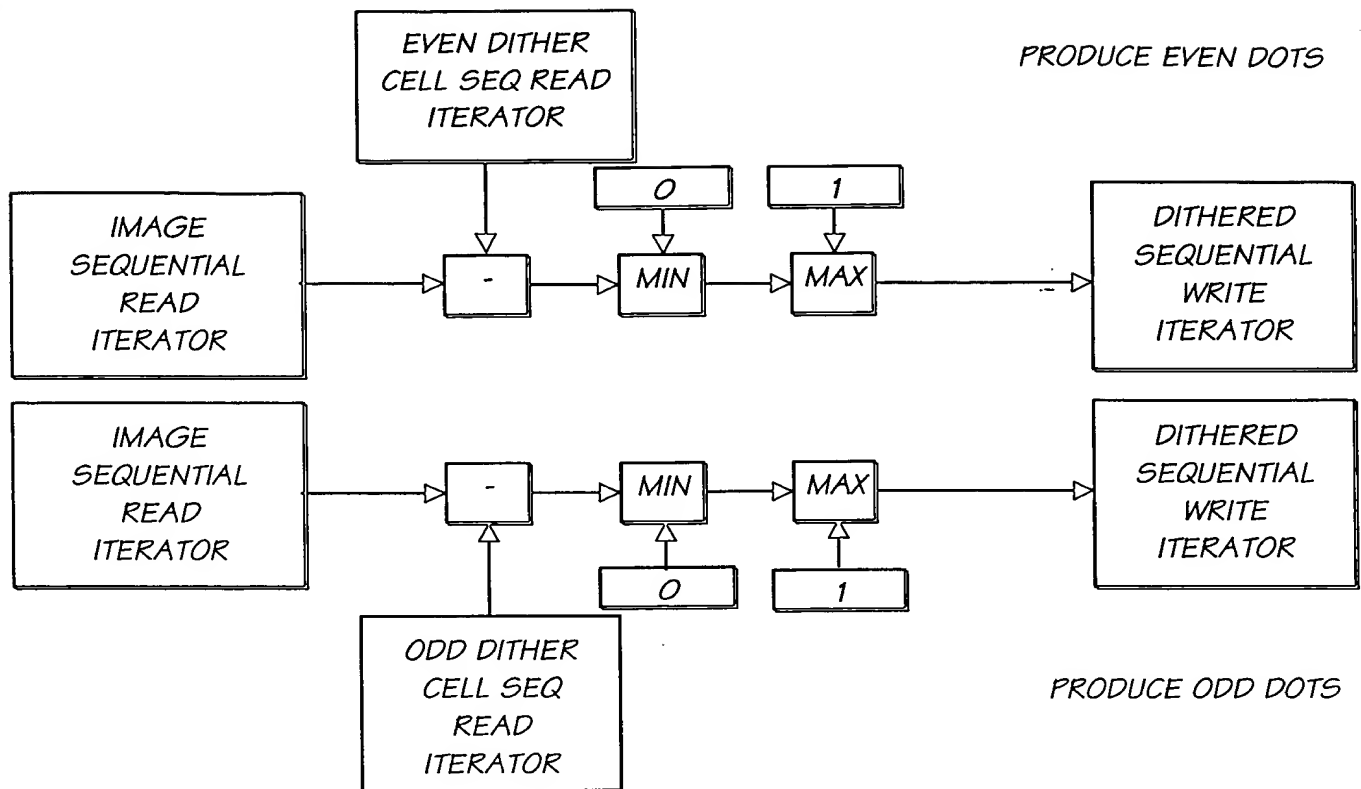


FIG. 122

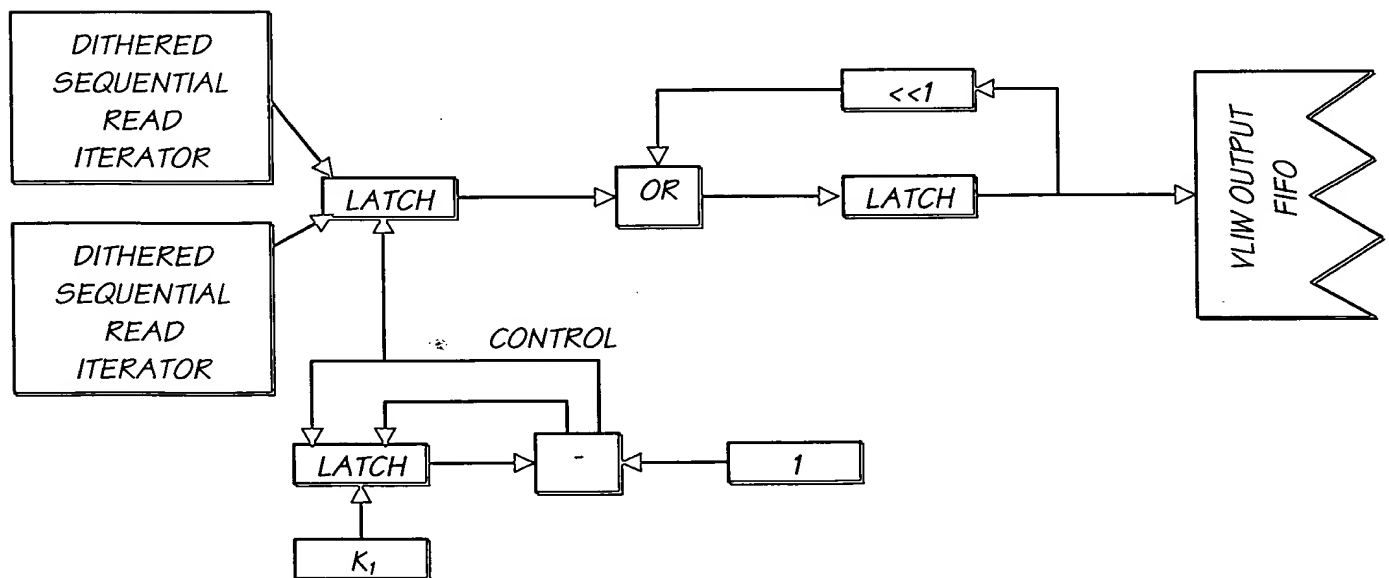


FIG. 123

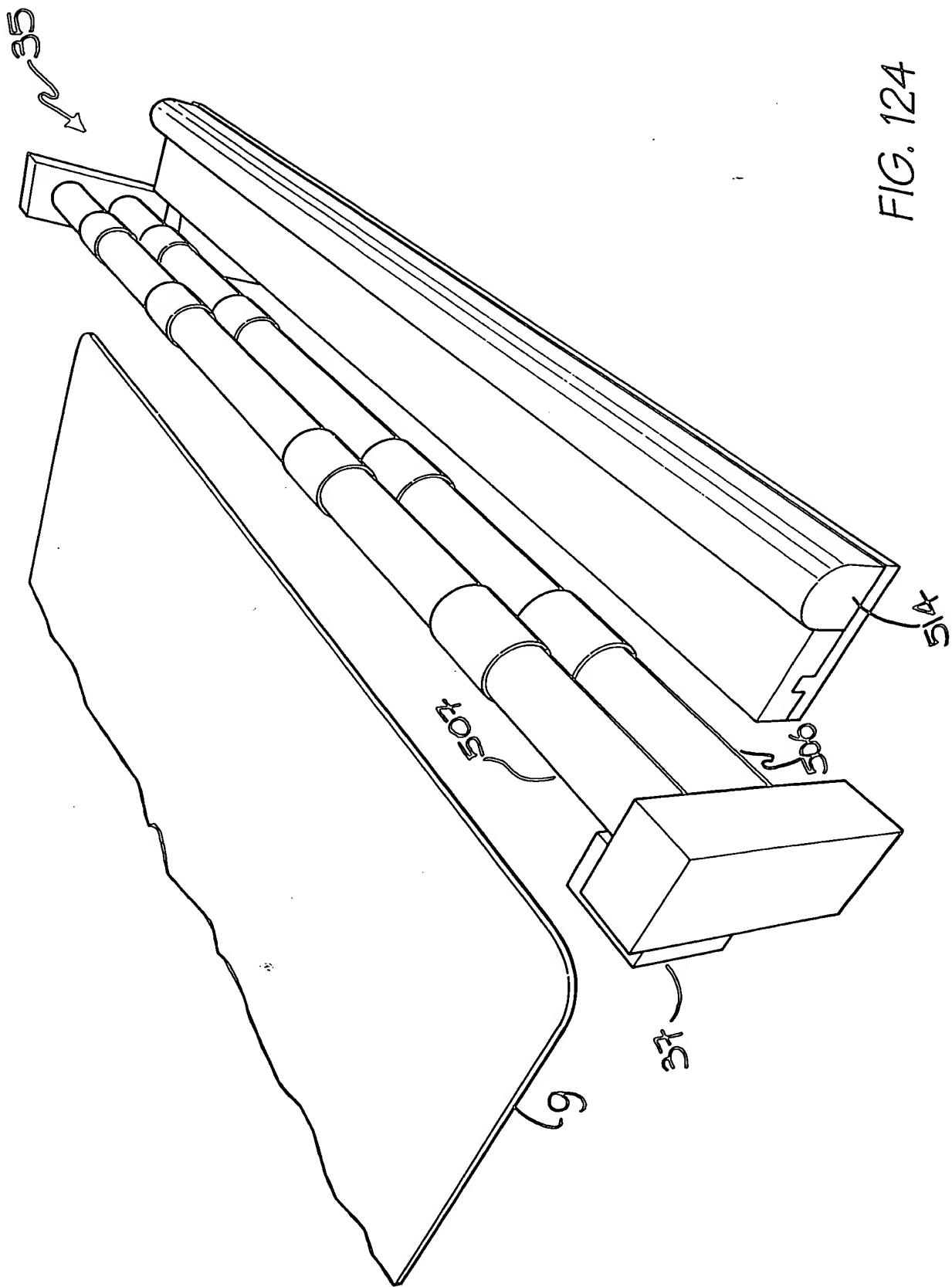


FIG. 124

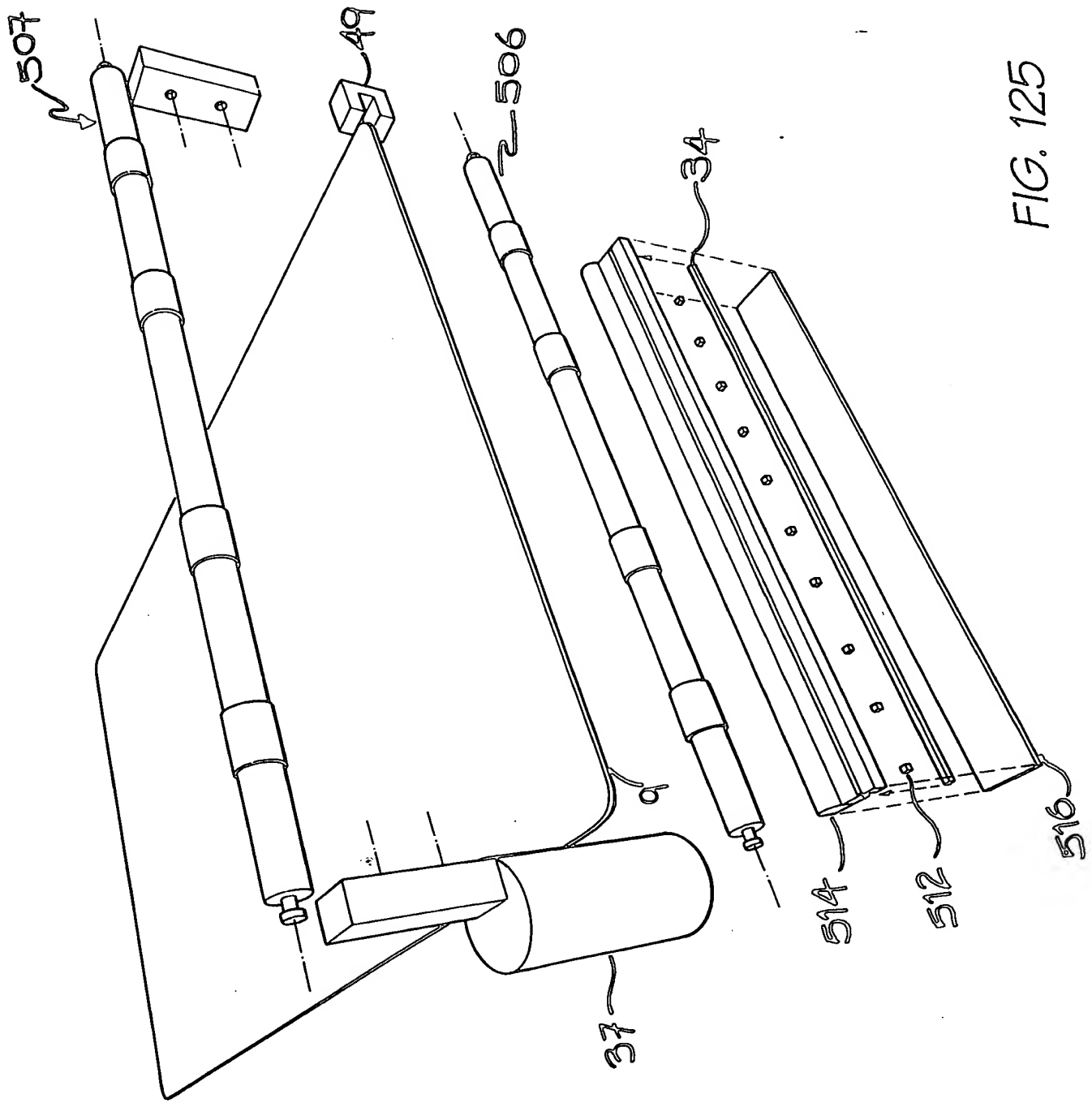


FIG. 125

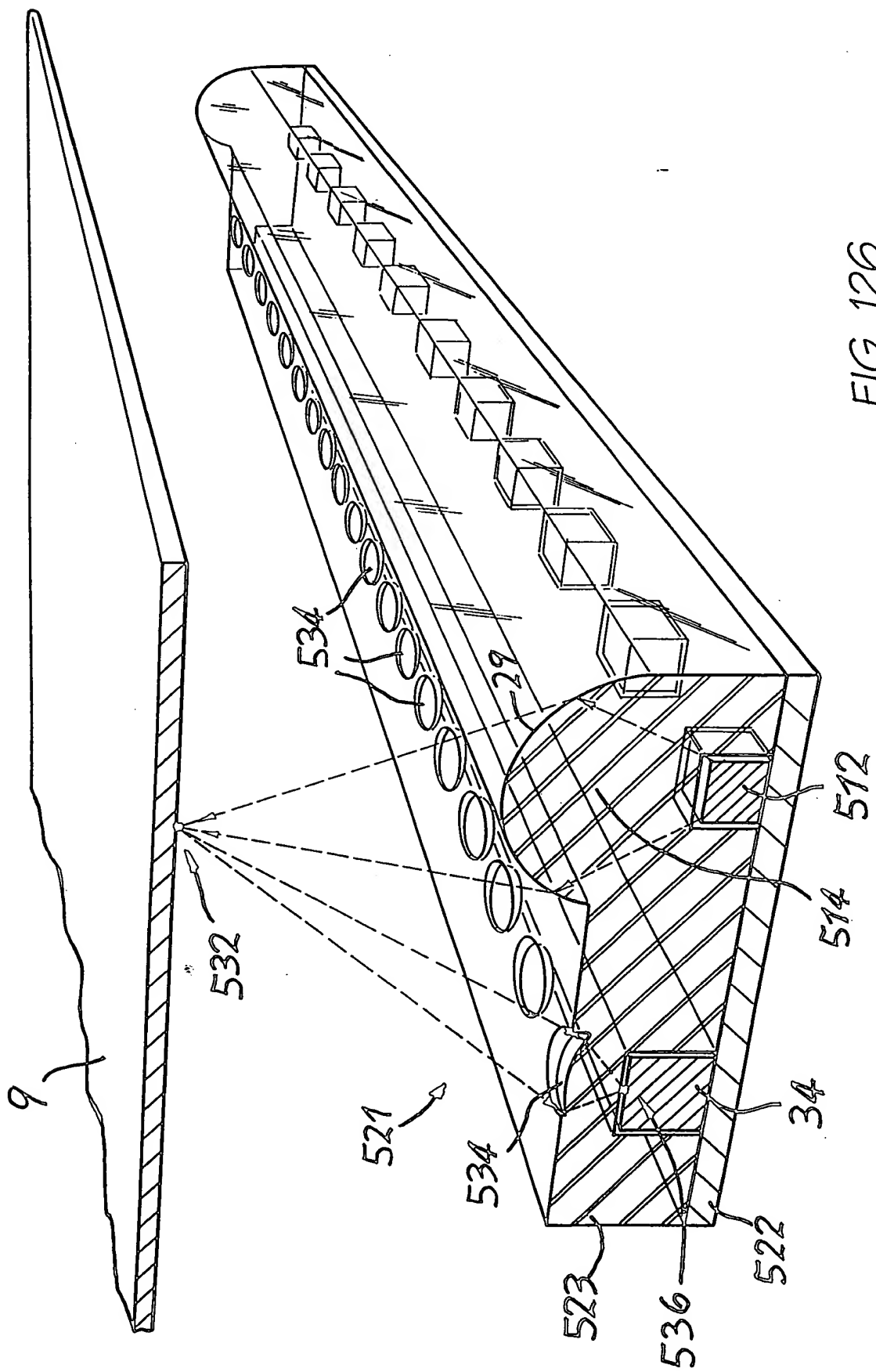
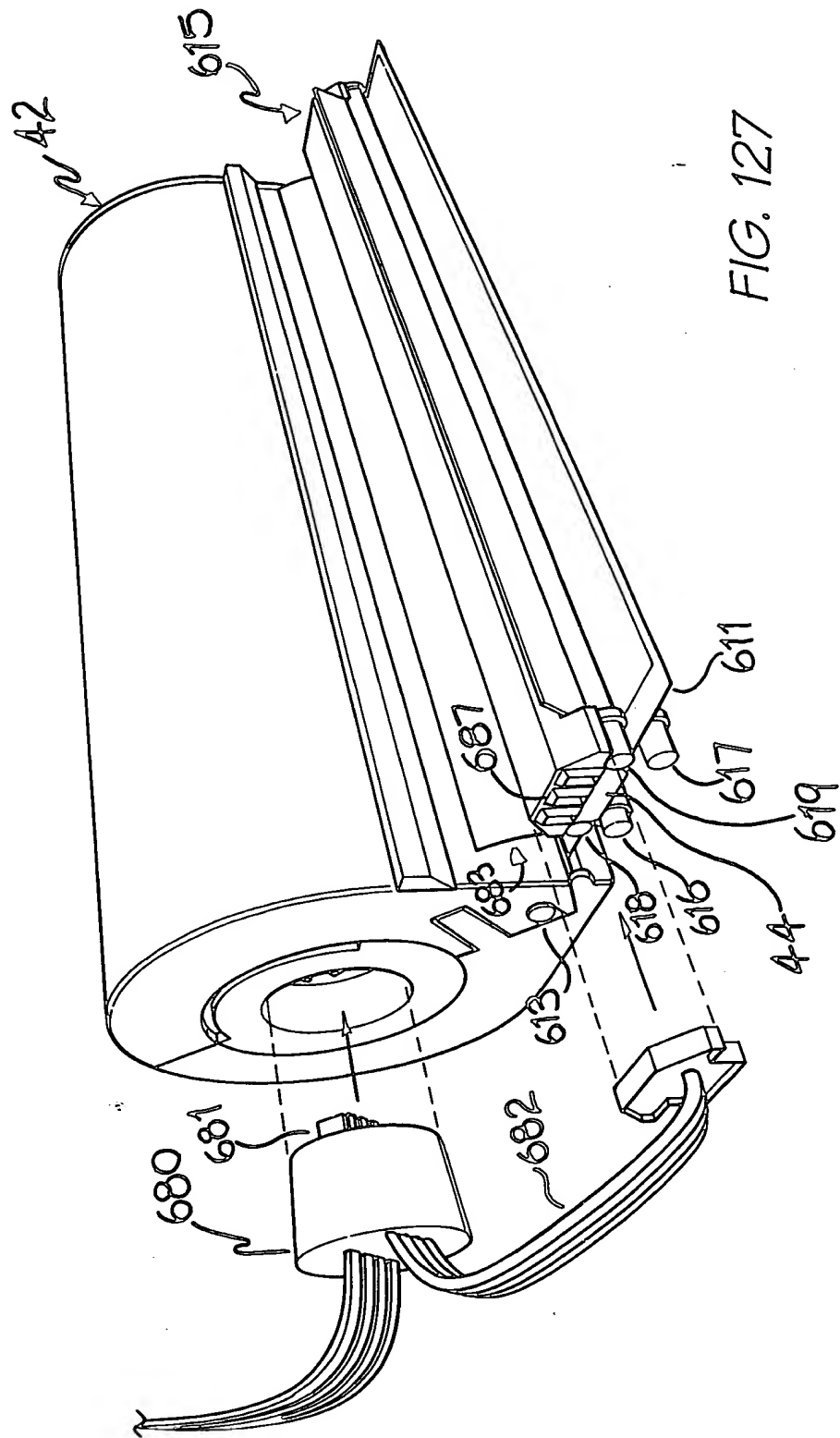


FIG. 126



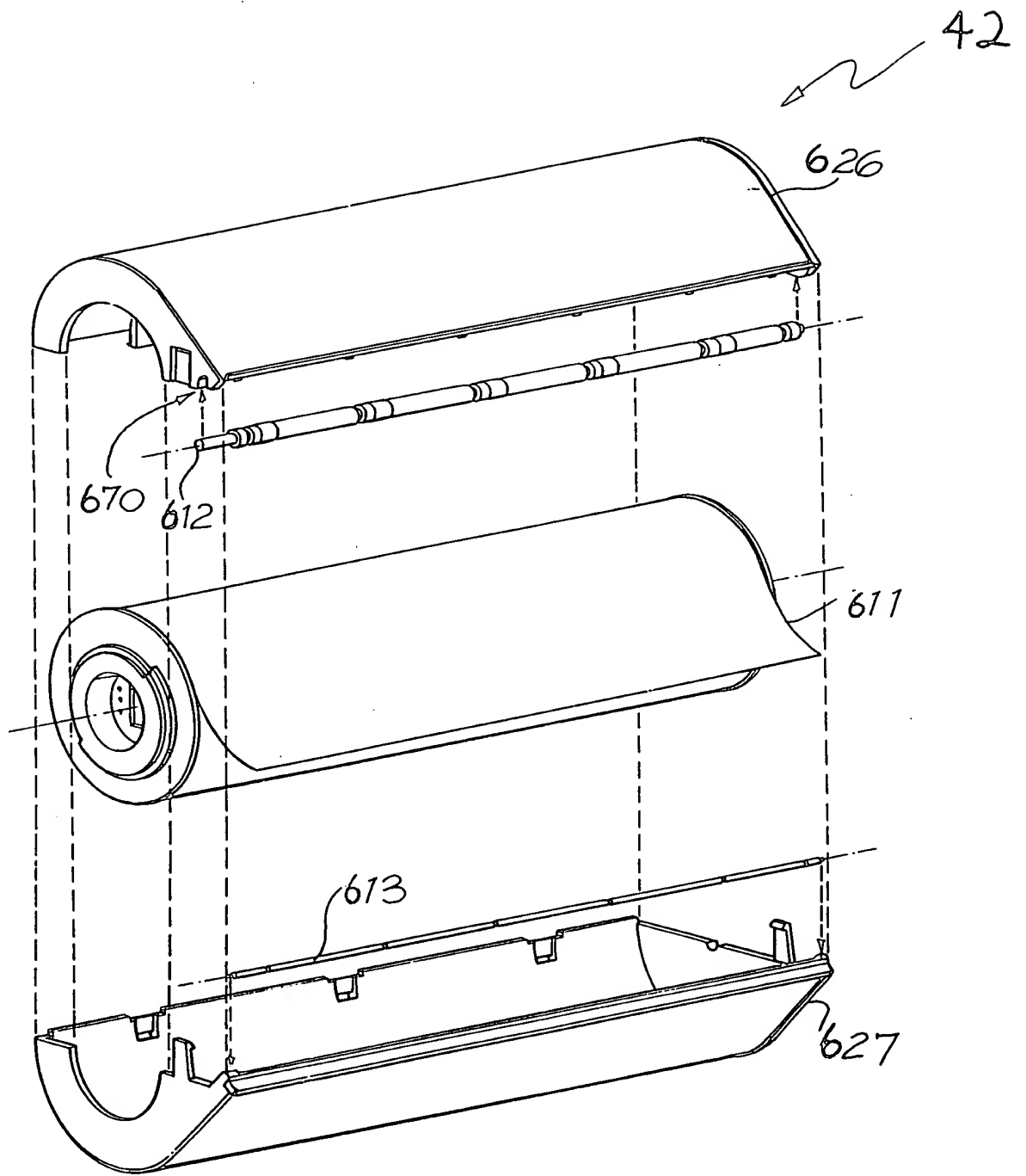


FIG. 128

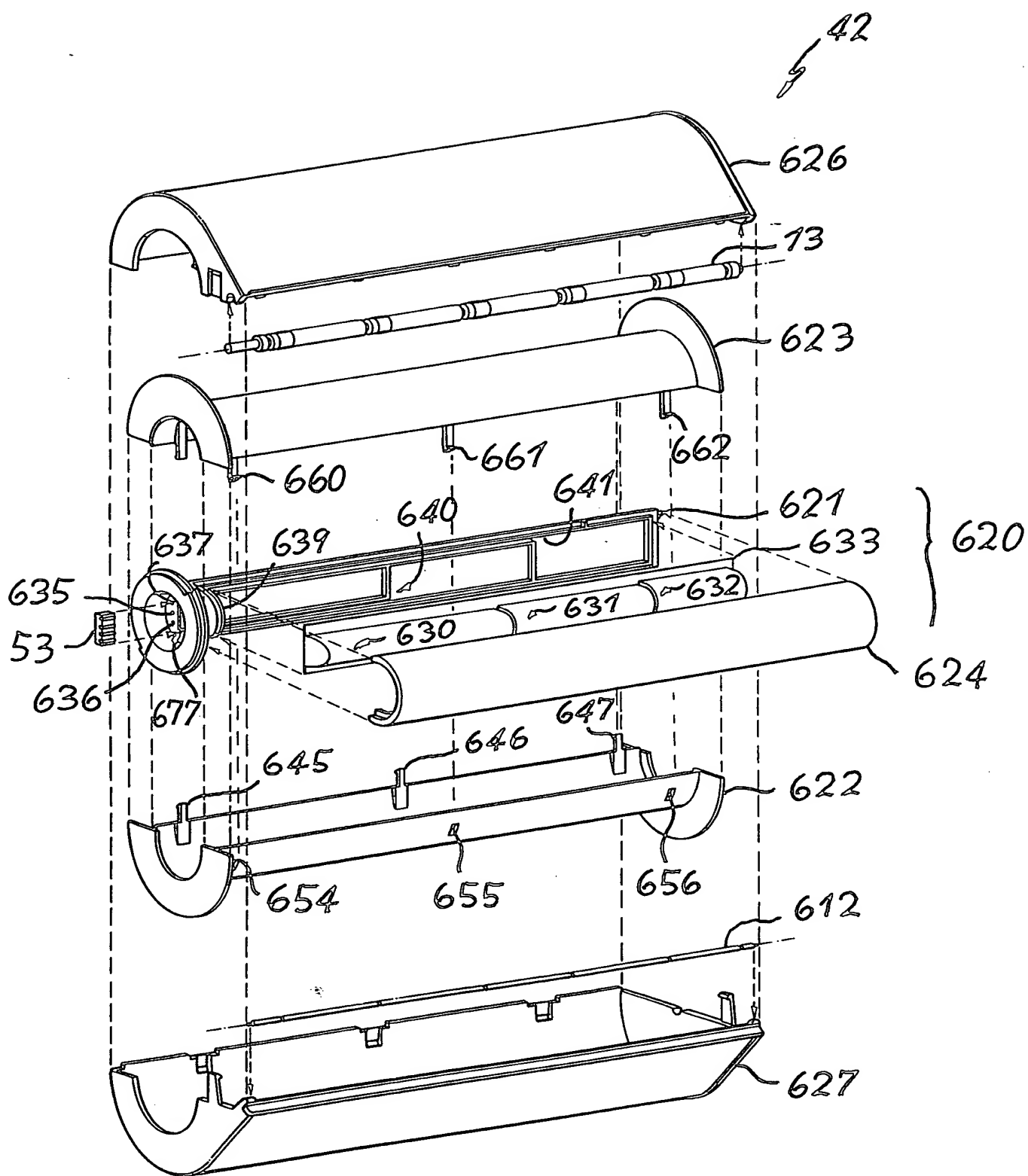


FIG. 129

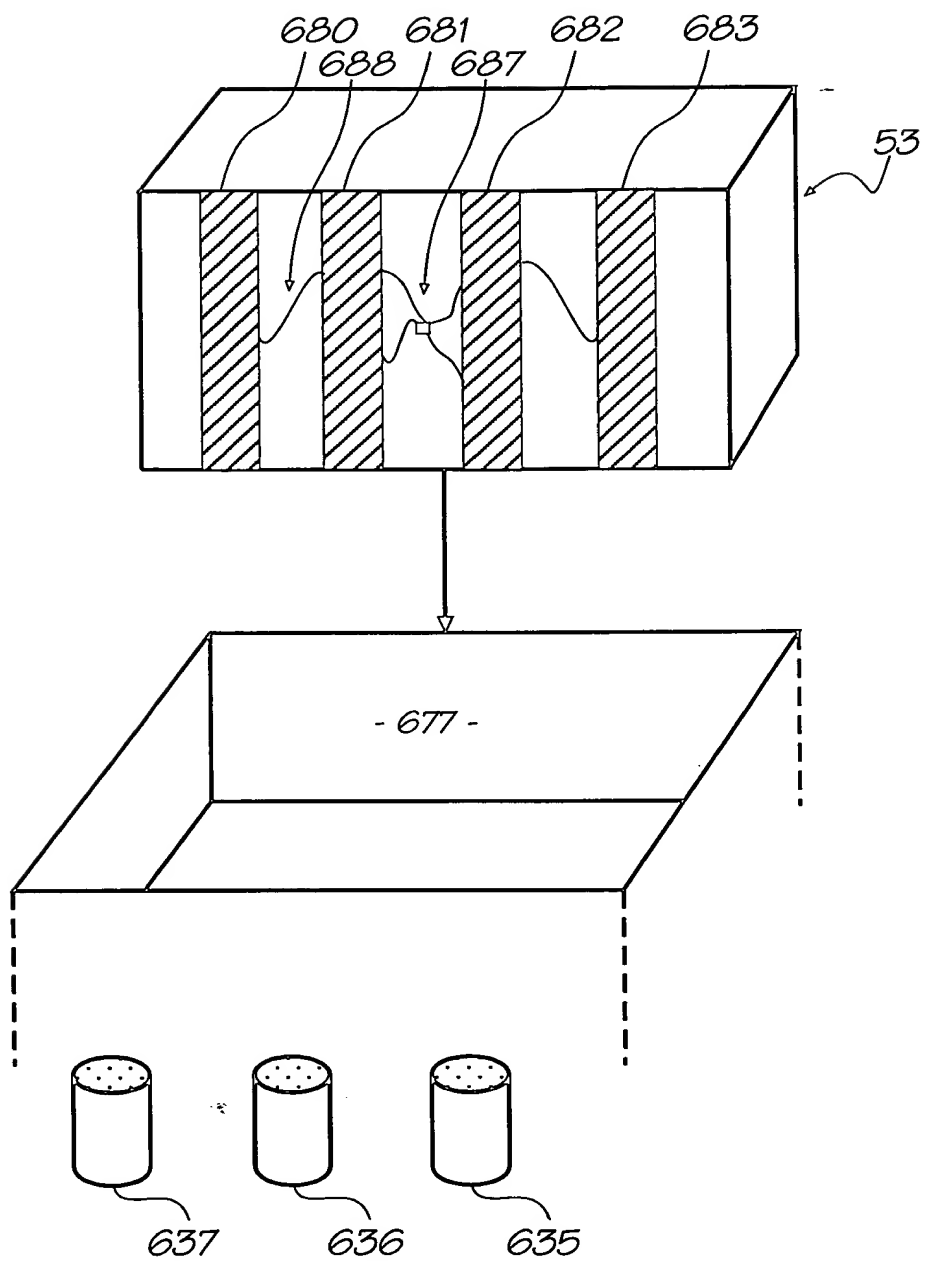


FIG. 130

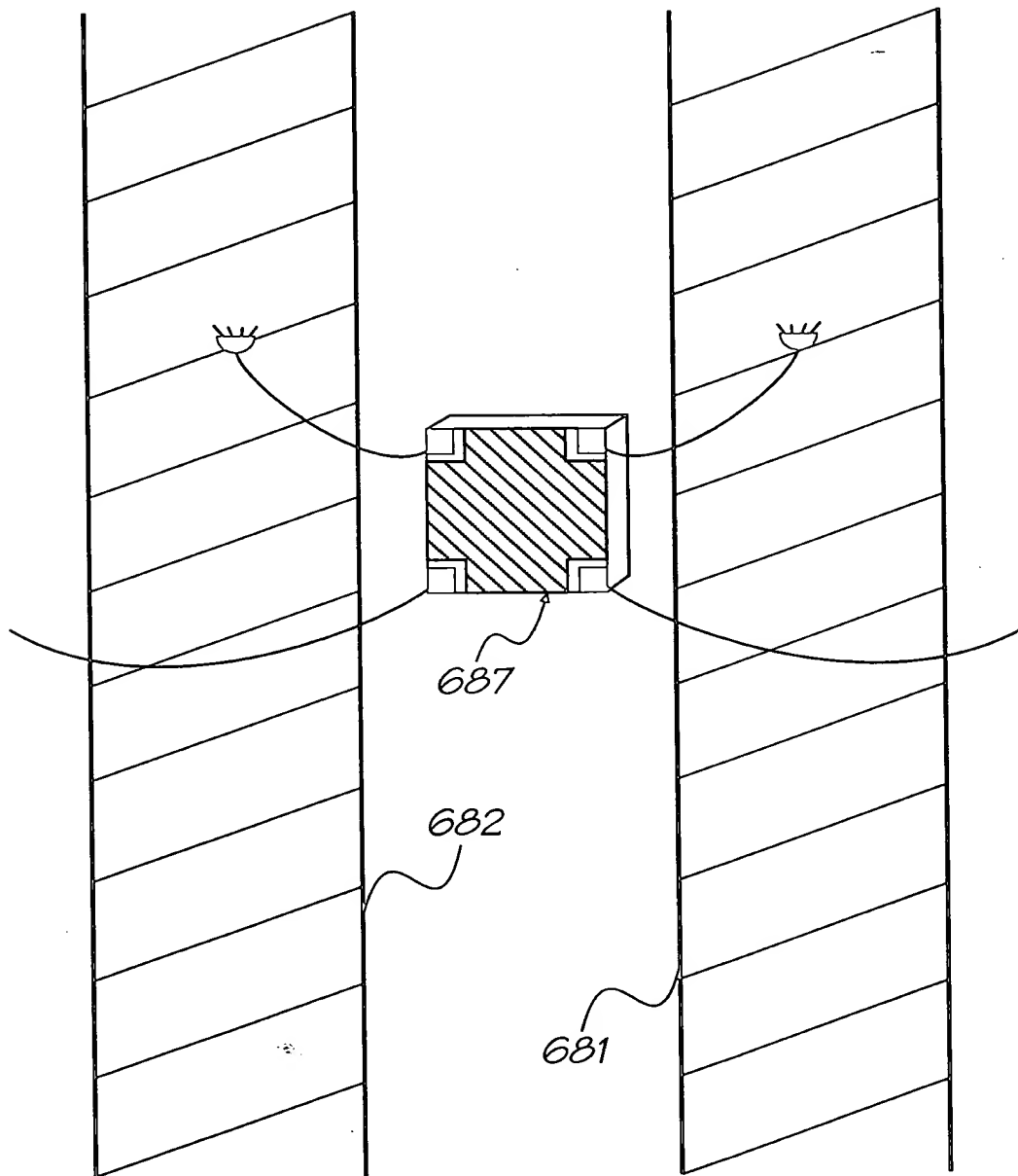


FIG. 131

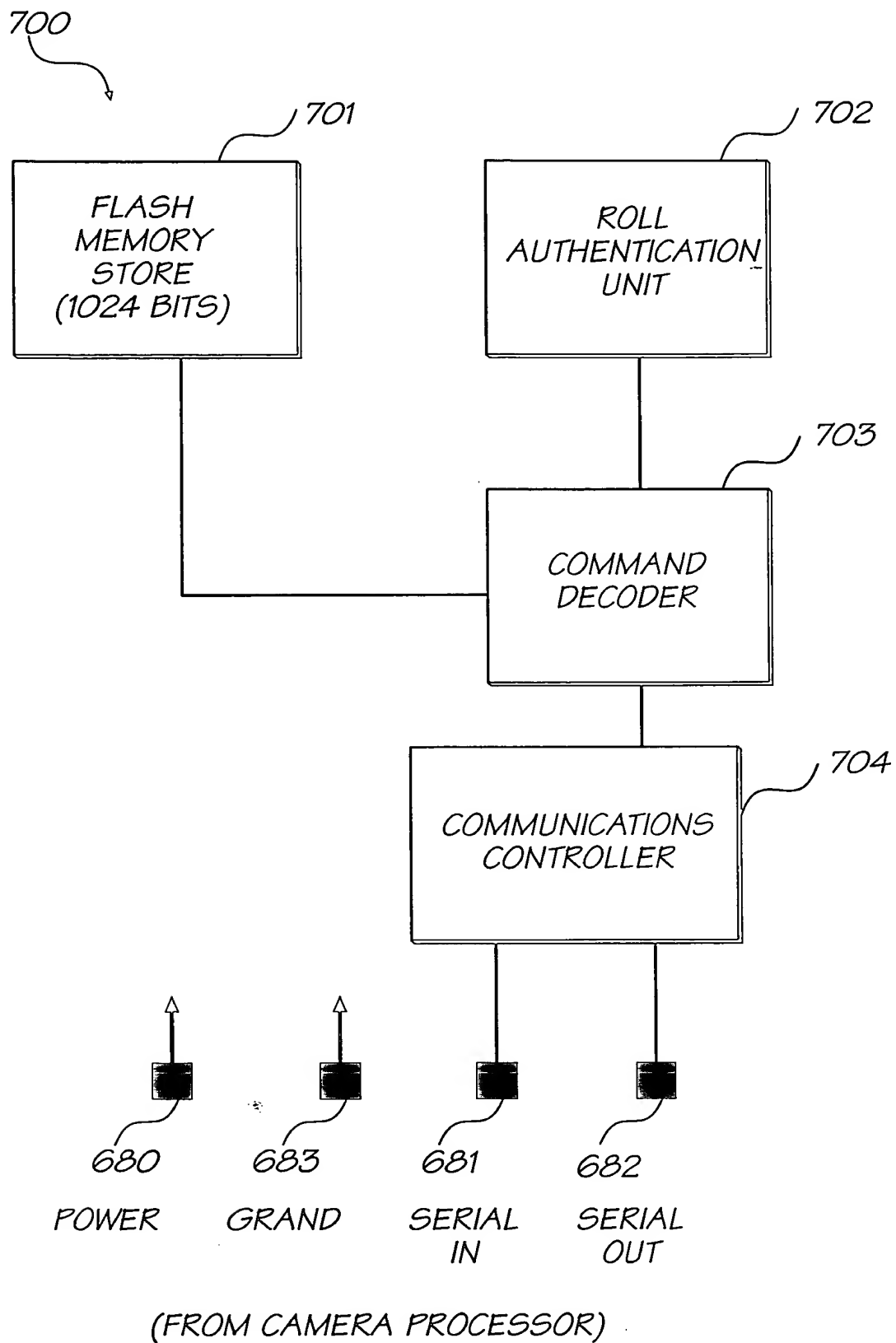
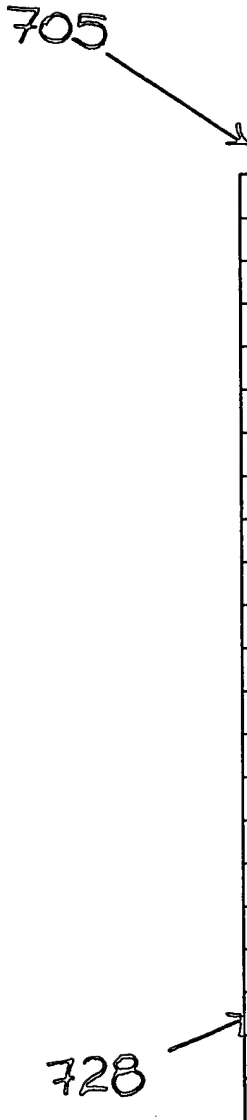


FIG. 132

705



Data Type	Bits
Factory code	16
Batch number	32
Serial number	48
Manufacturing date	16
Media length	24
Media type	8
Preprinted media length	16
Cyan ink viscosity	8
Magenta ink viscosity	8
Yellow ink viscosity	8
Cyan drop volume	8
Magenta drop volume	8
Yellow drop volume	8
Cyan ink color	24
Magenta ink color	24
Yellow ink color	24
Remaining-media length indicator	16
Authentication key	128
Copyrightable bit pattern	512
Reserved for camera use	88
Total	1024

728

FIG. 133

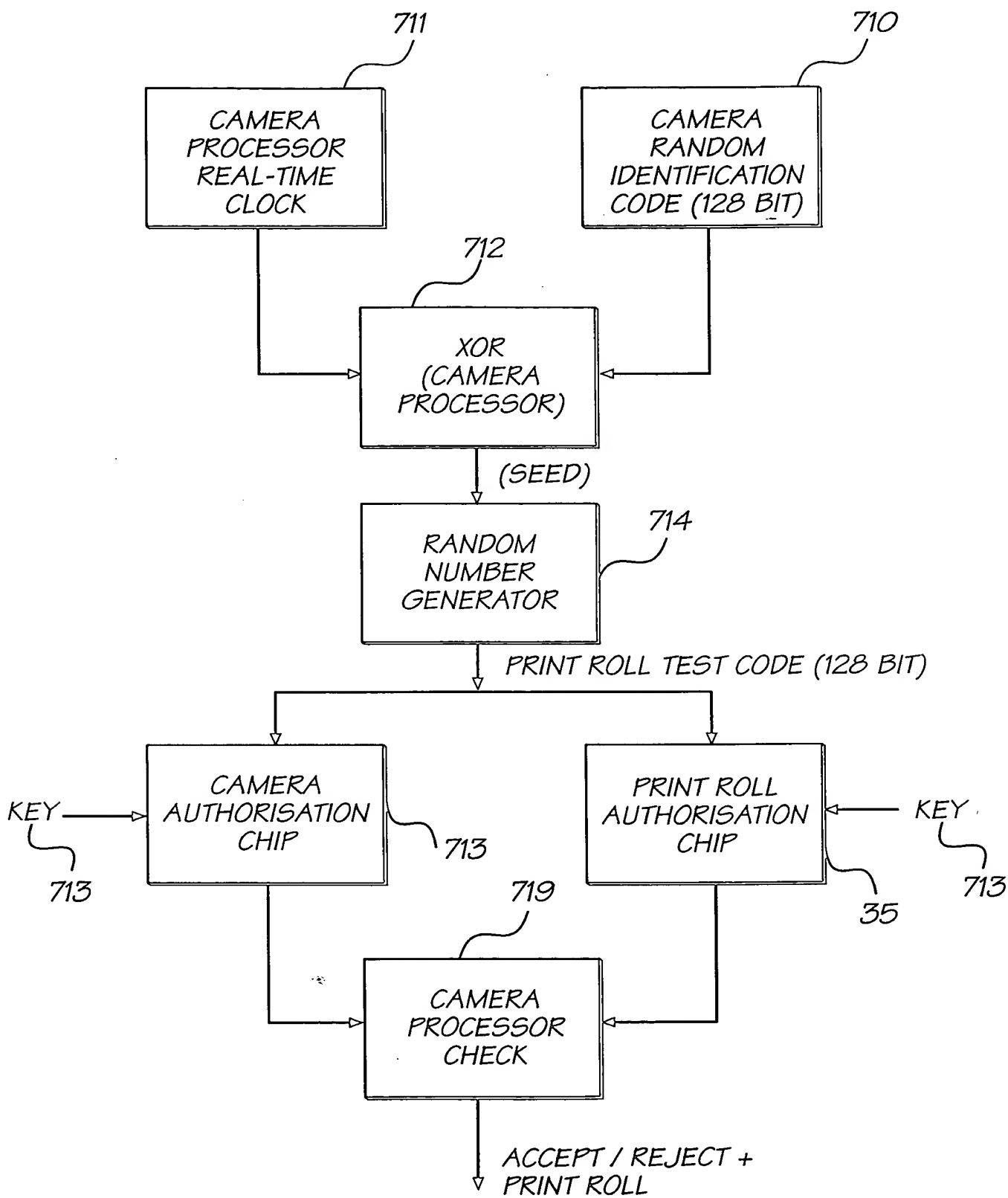


FIG. 134

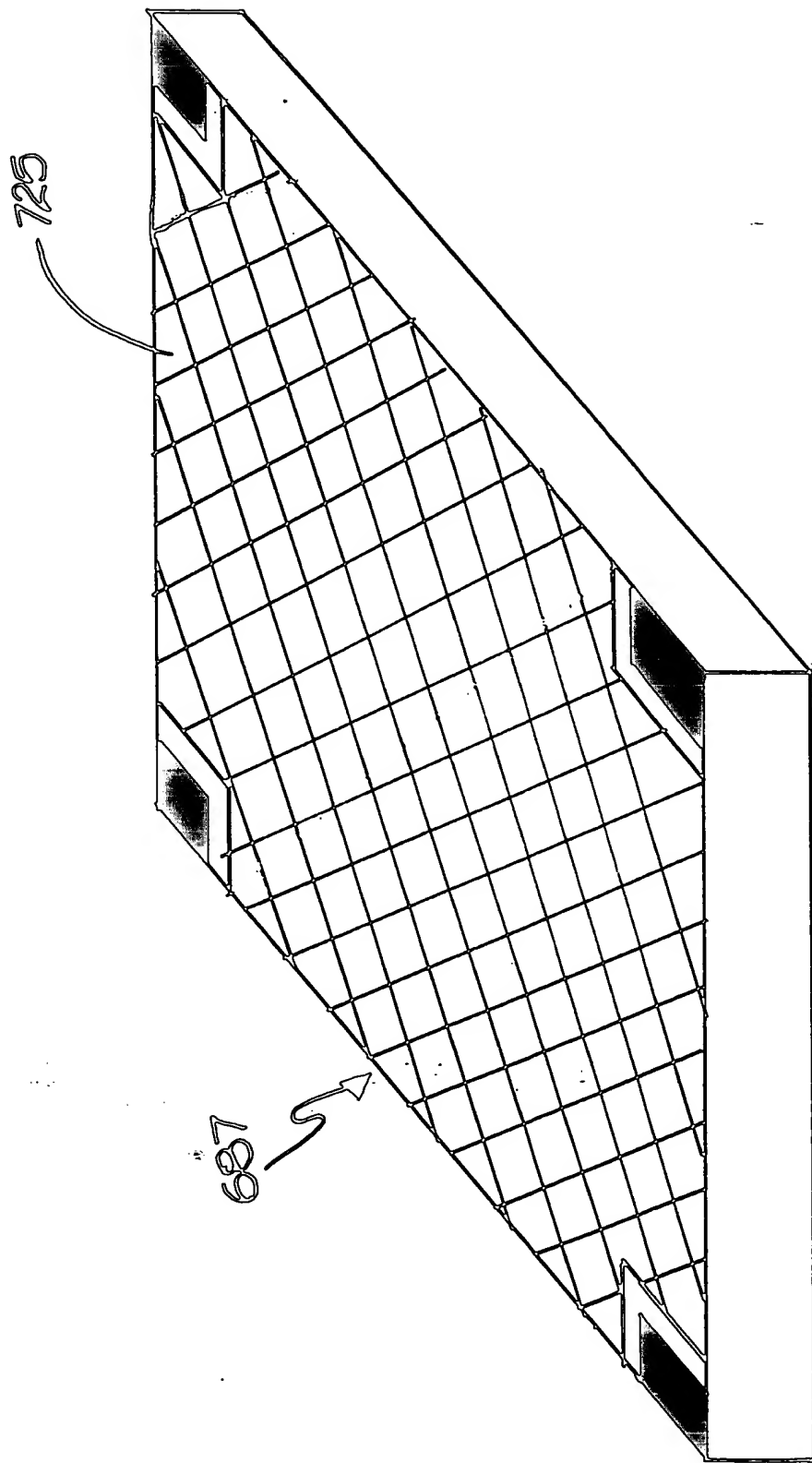
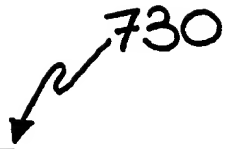


FIG. 135



Data Type	Bits
Manufacturer code	16
Batch number	32
Serial number	32
Manufacturing date	16
Print engine type	8
Print resolution	16
Print counter	16
Authentication test key (random)	128
Print roll Authentication key	128
Bit pattern	512
Spare for camera use	120
Total	1024

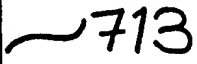


FIG. 136

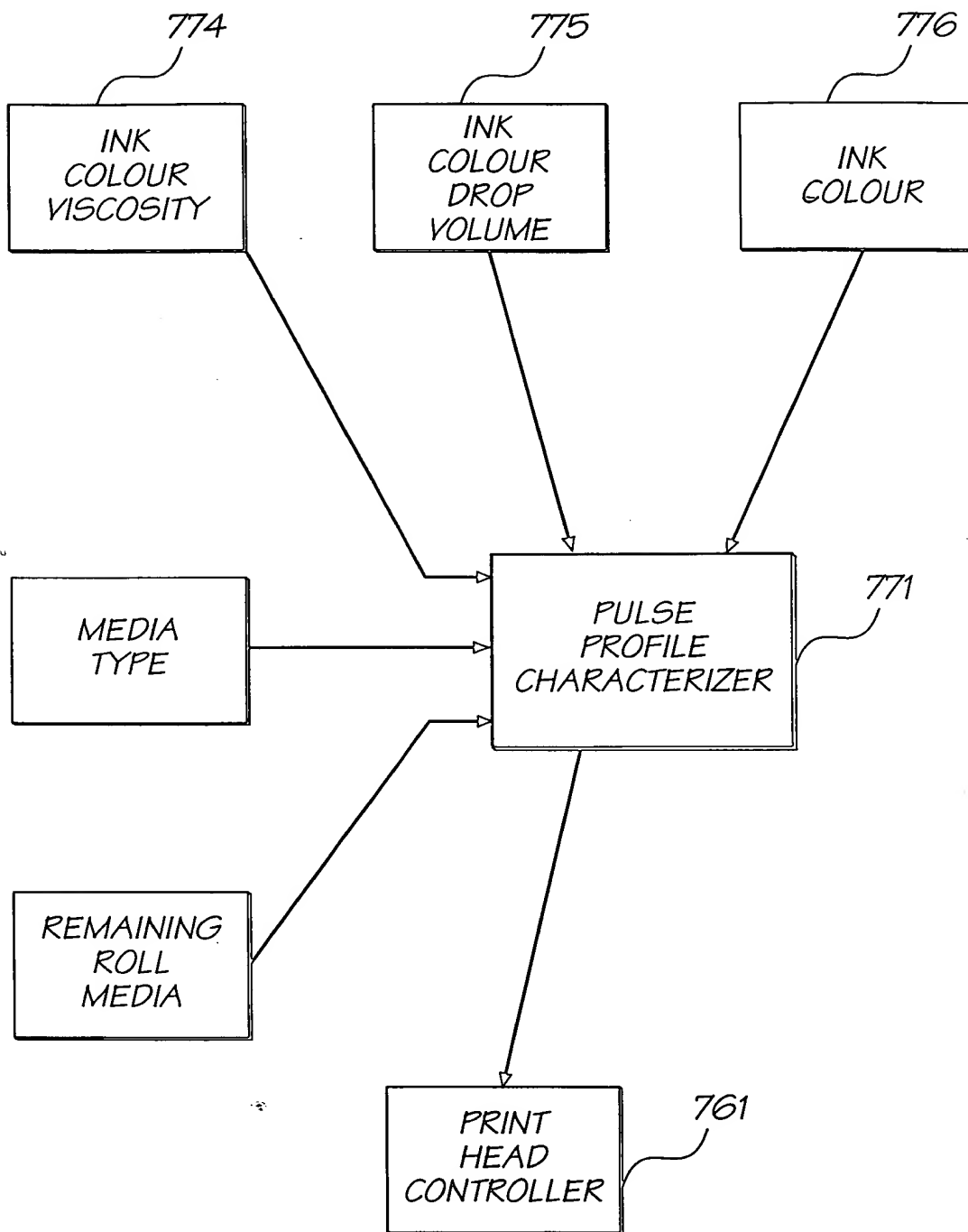


FIG. 137

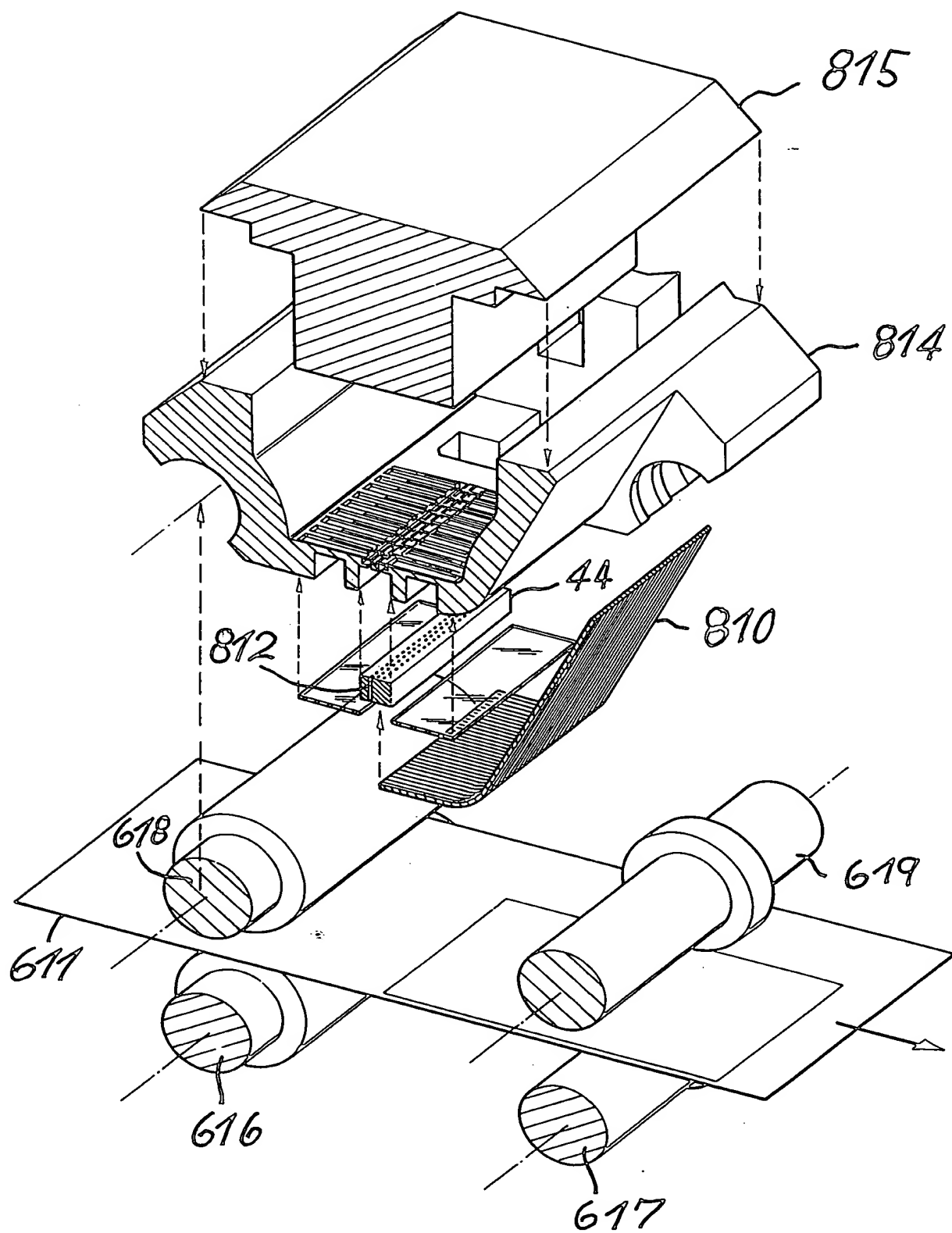


FIG. 138

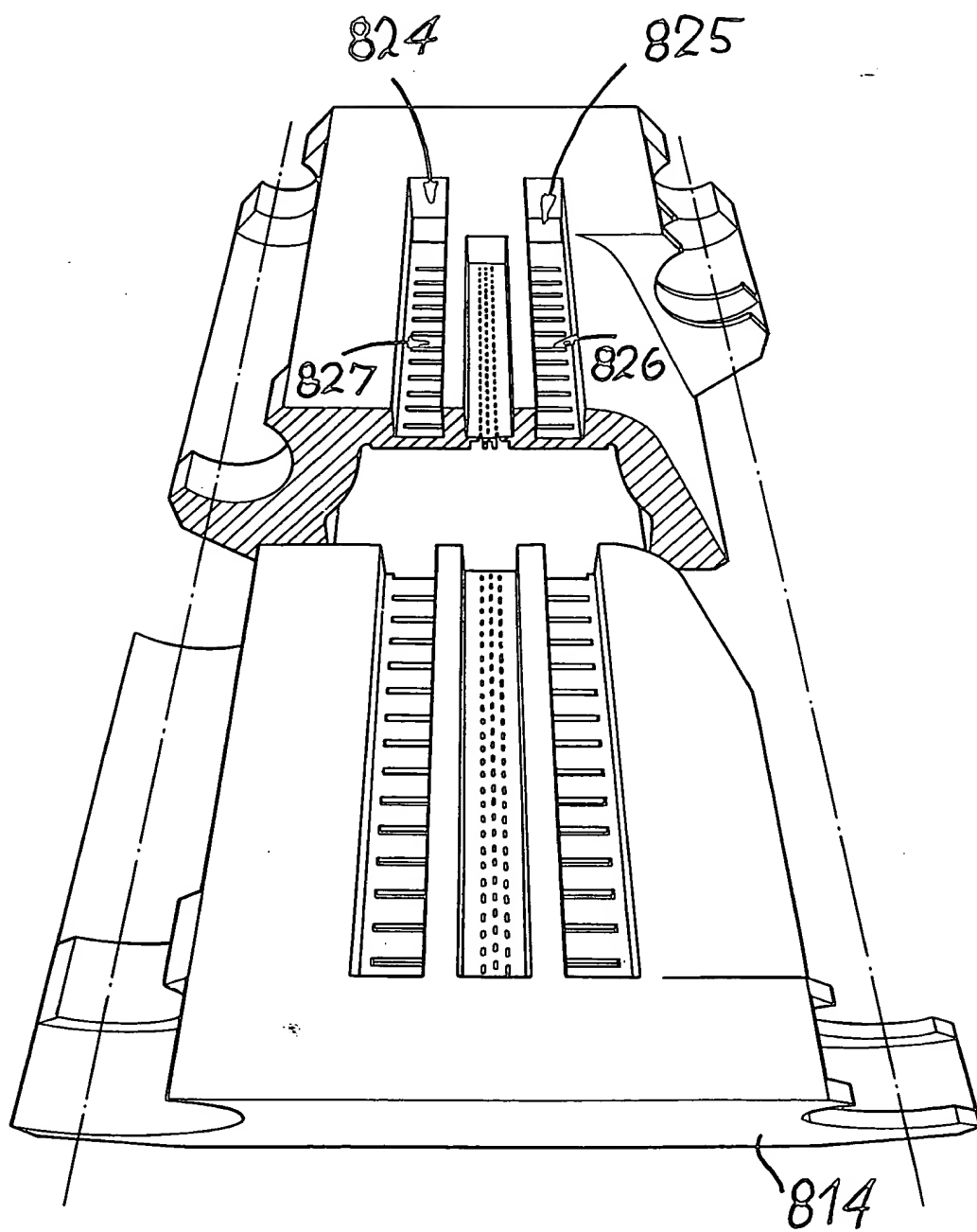


FIG. 139

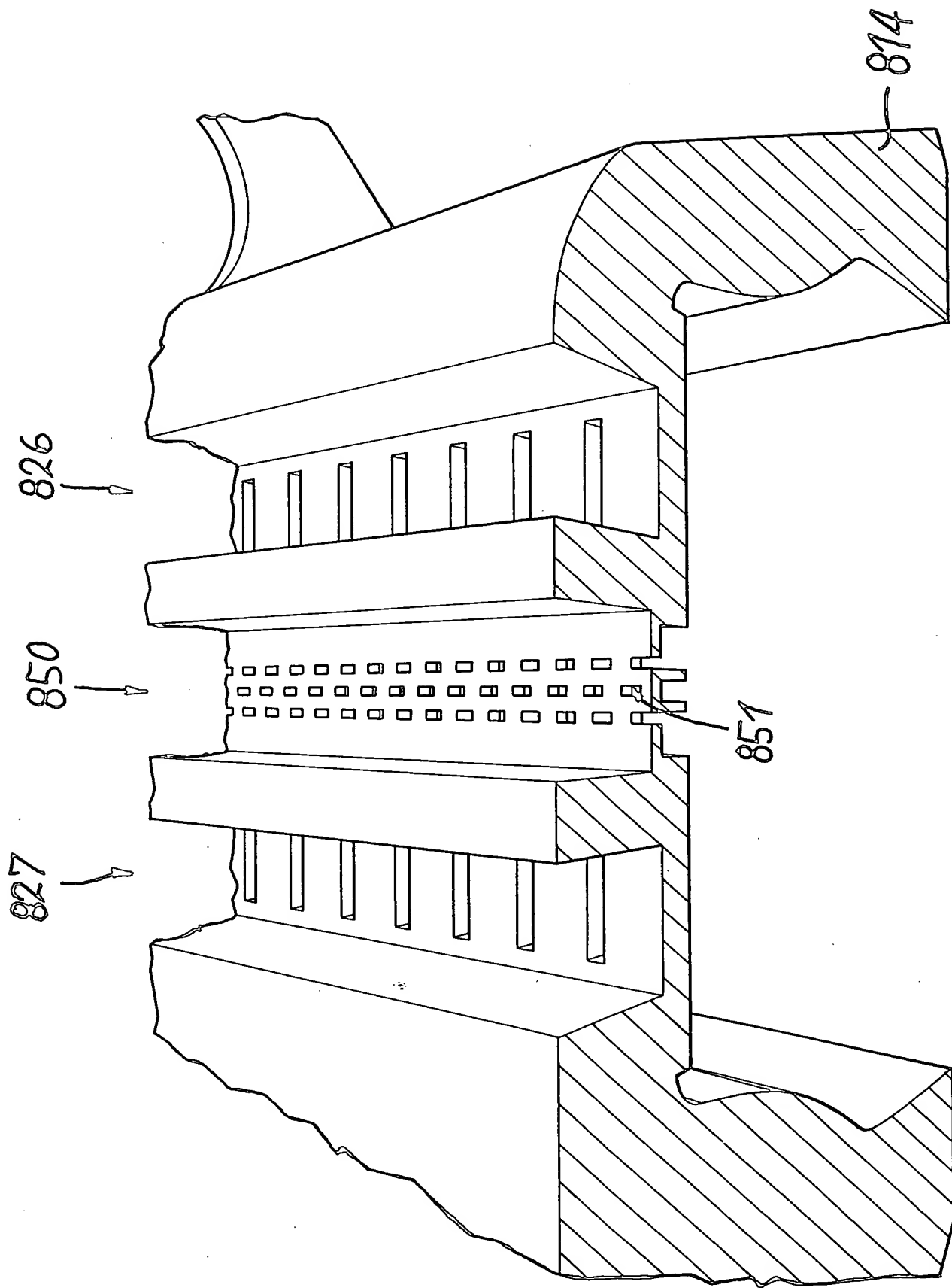


FIG. 140

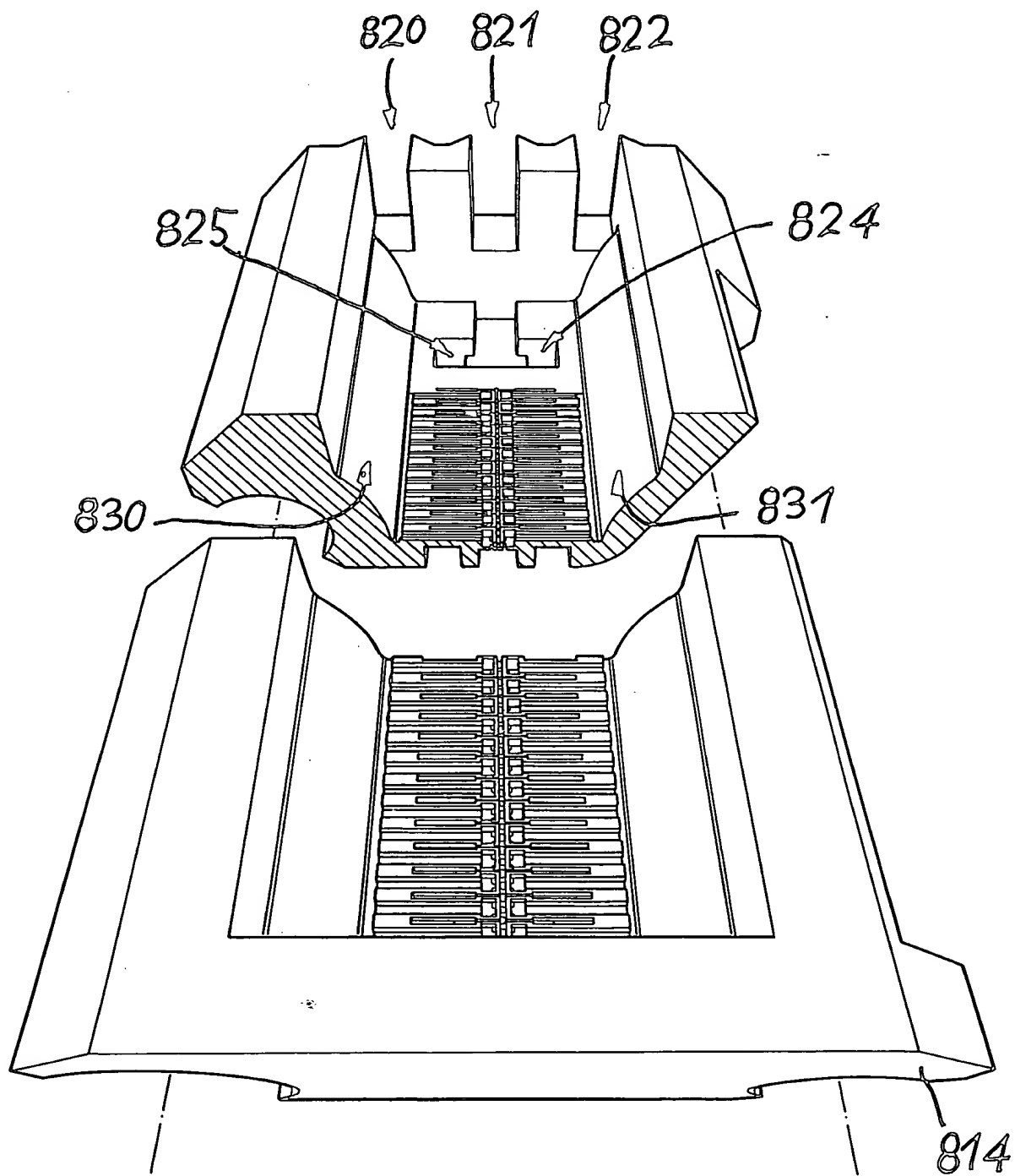


FIG. 141

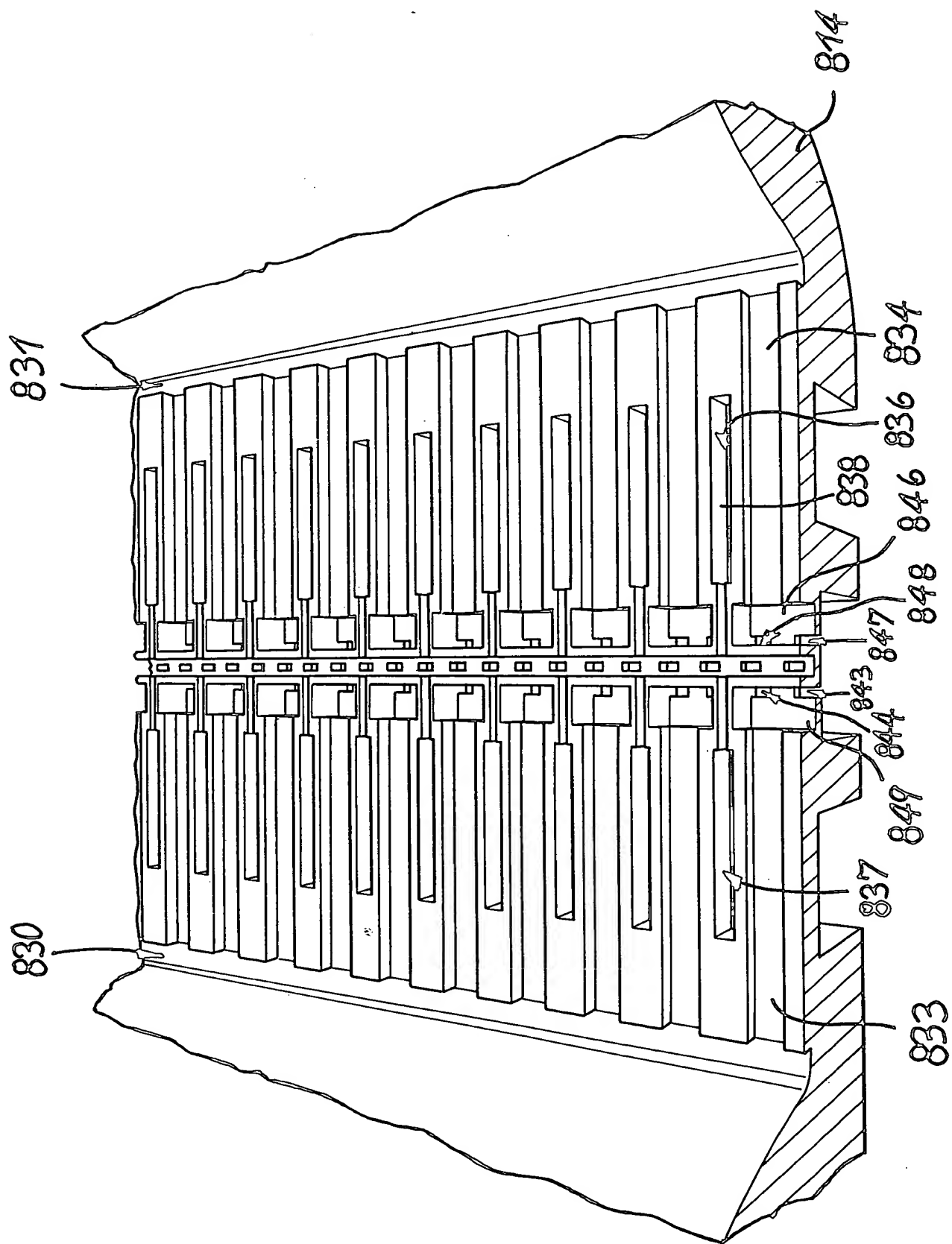


FIG. 142

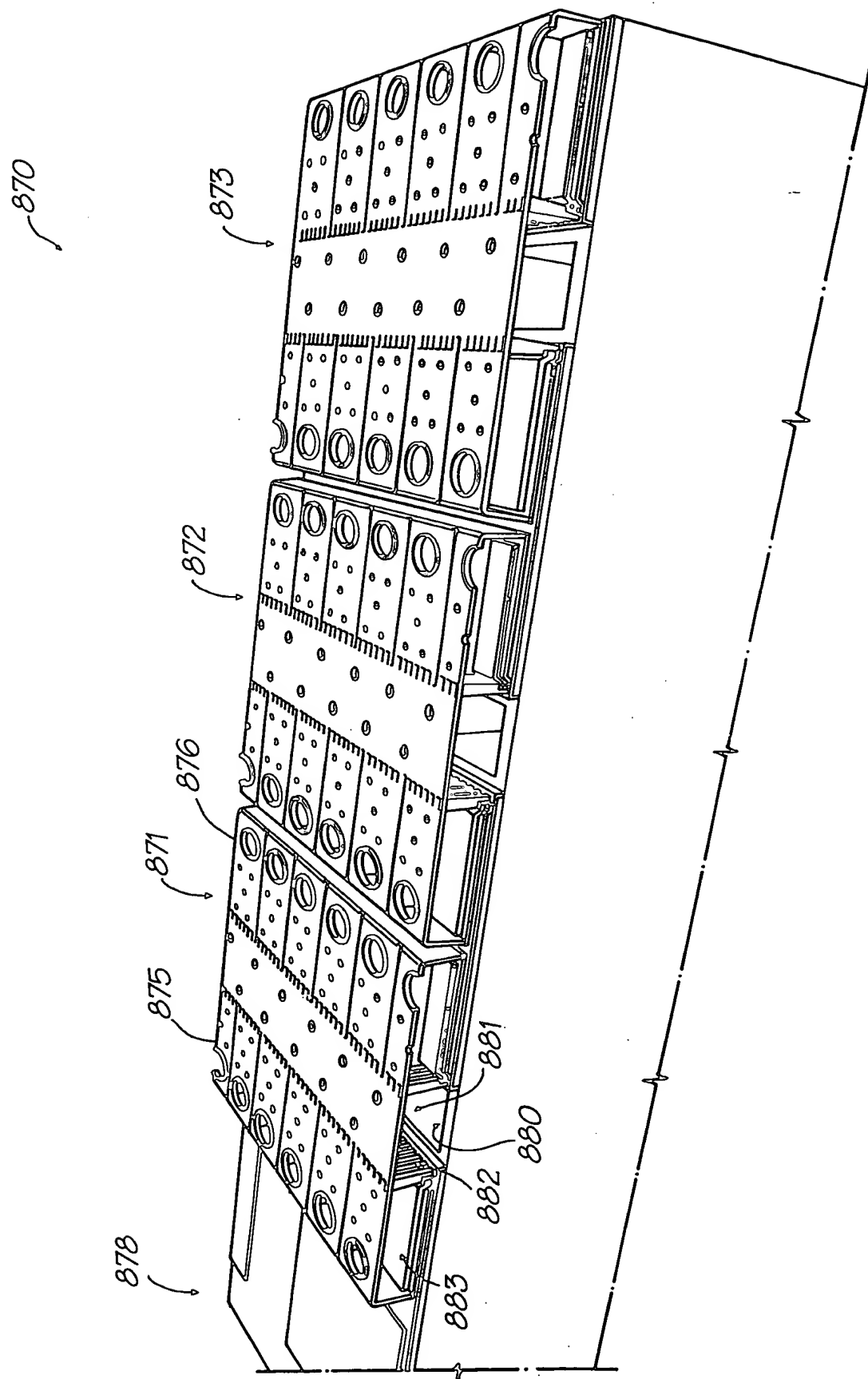


FIG. 143

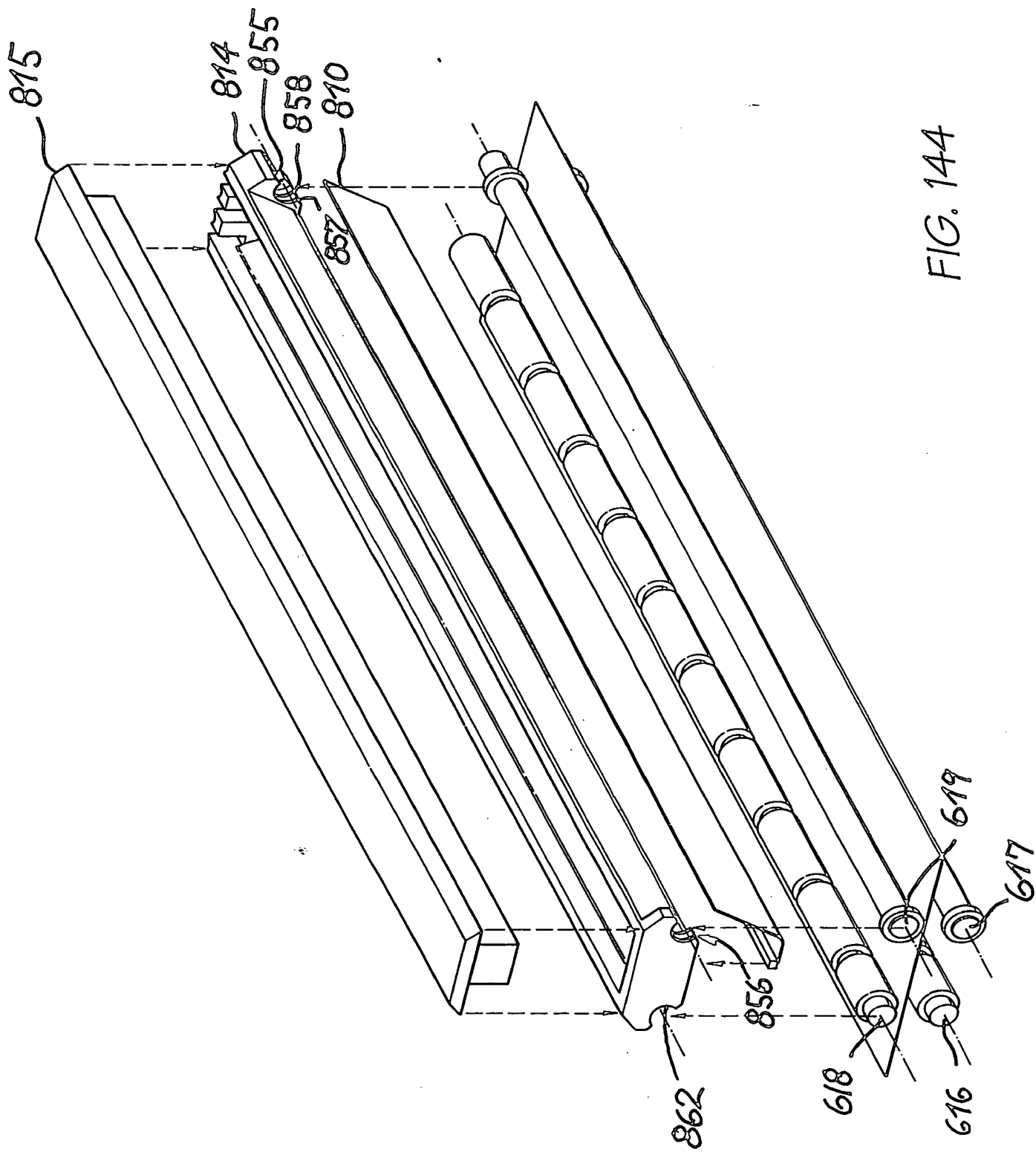


FIG. 144

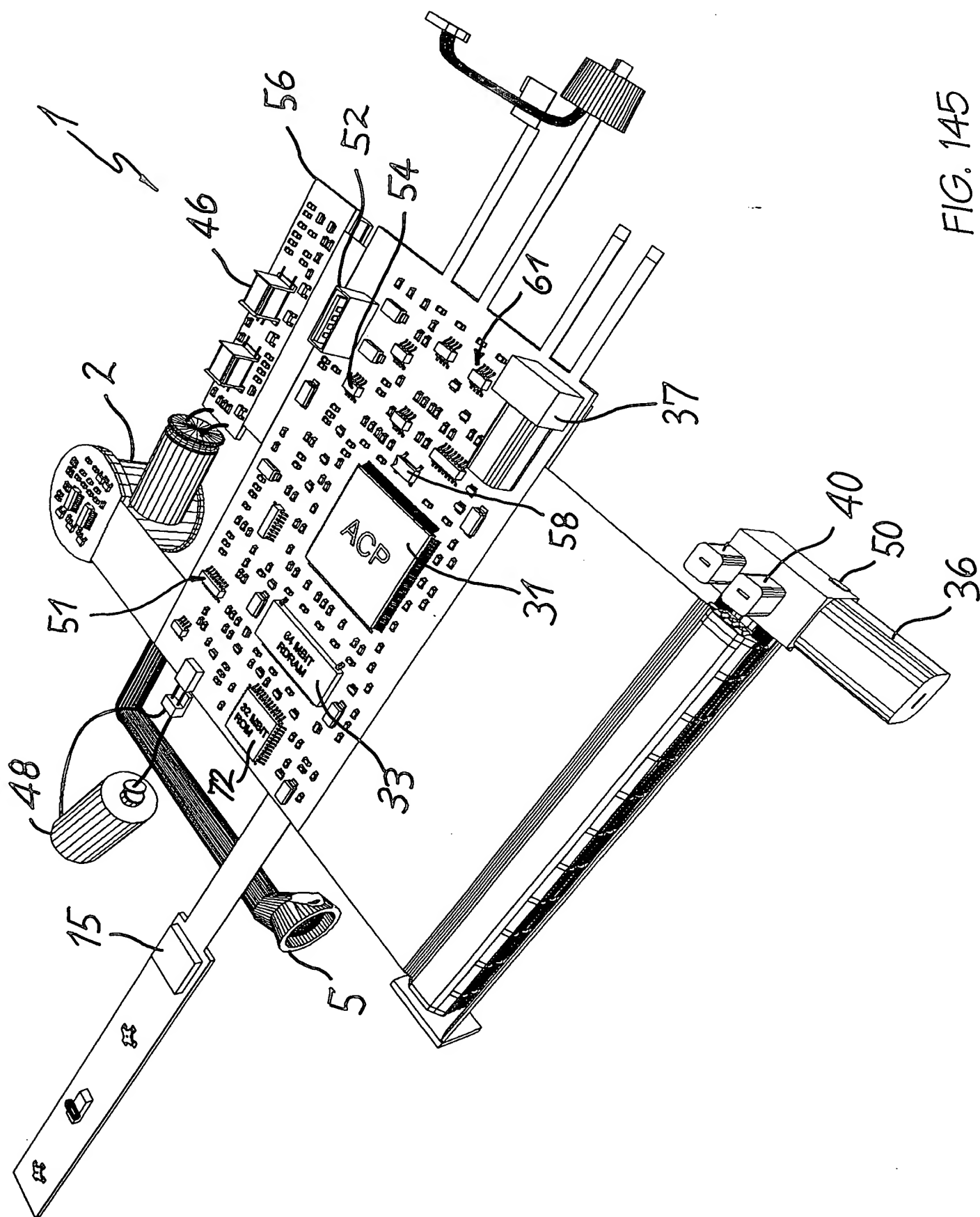


FIG. 145

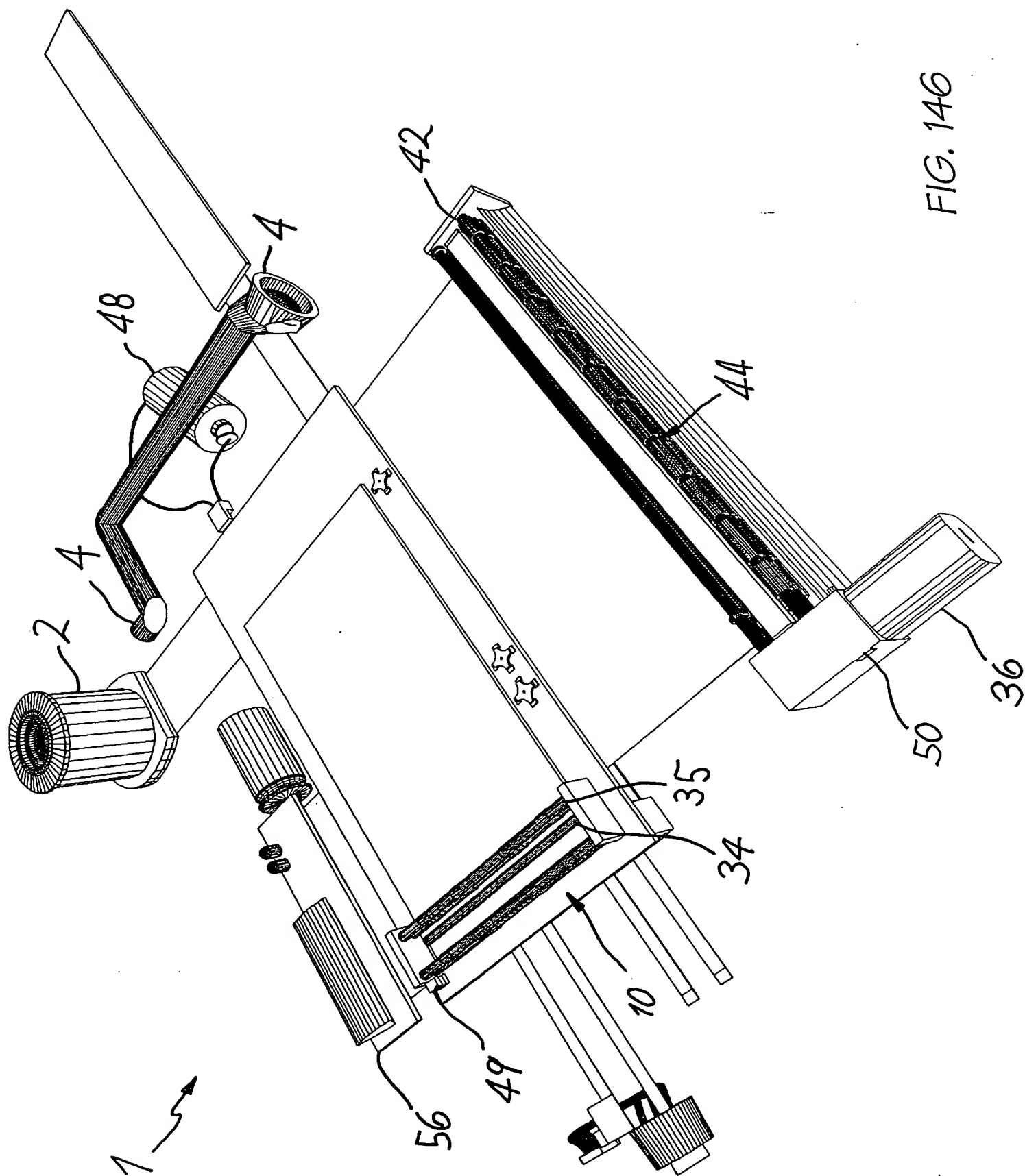


FIG. 146

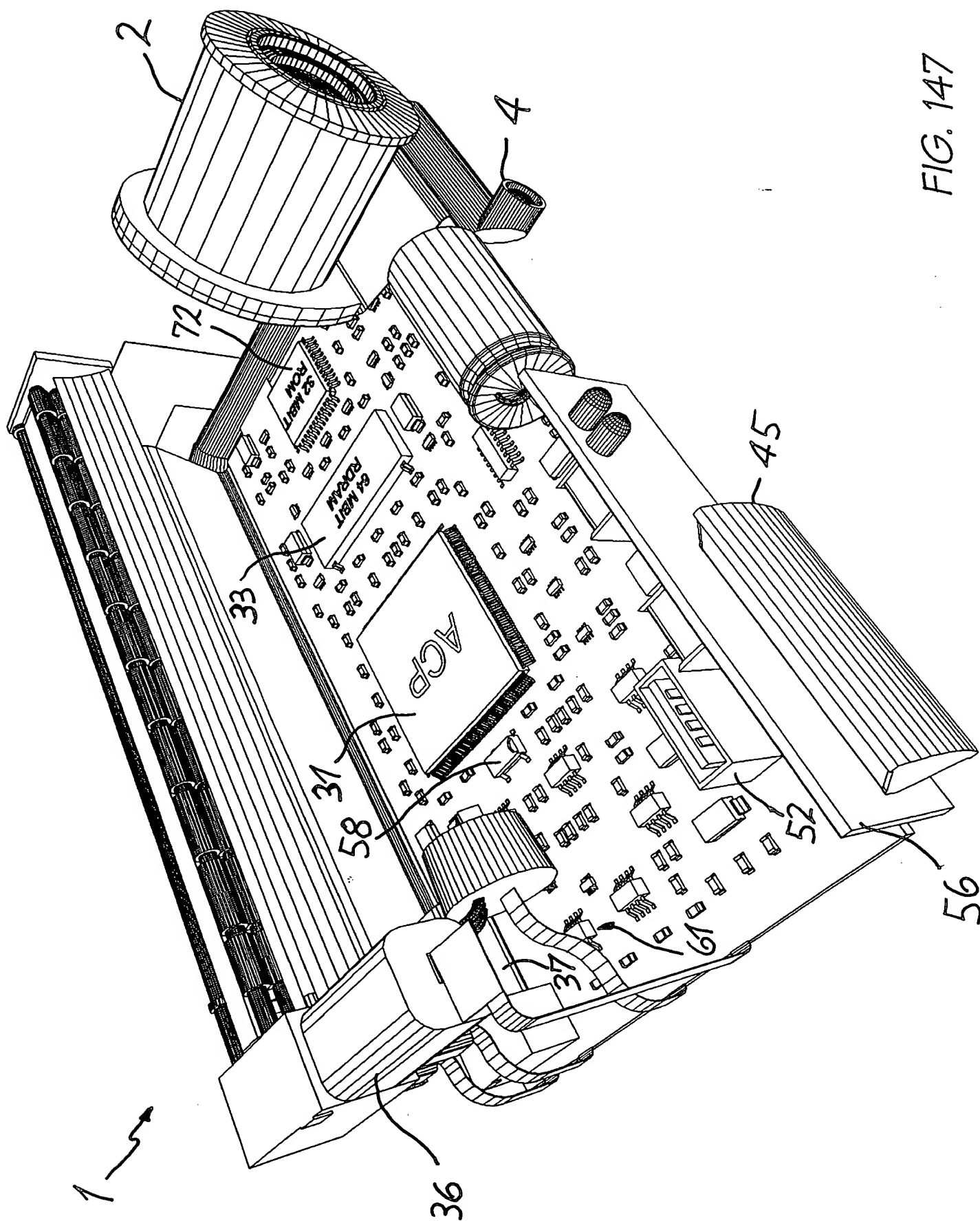


FIG. 147

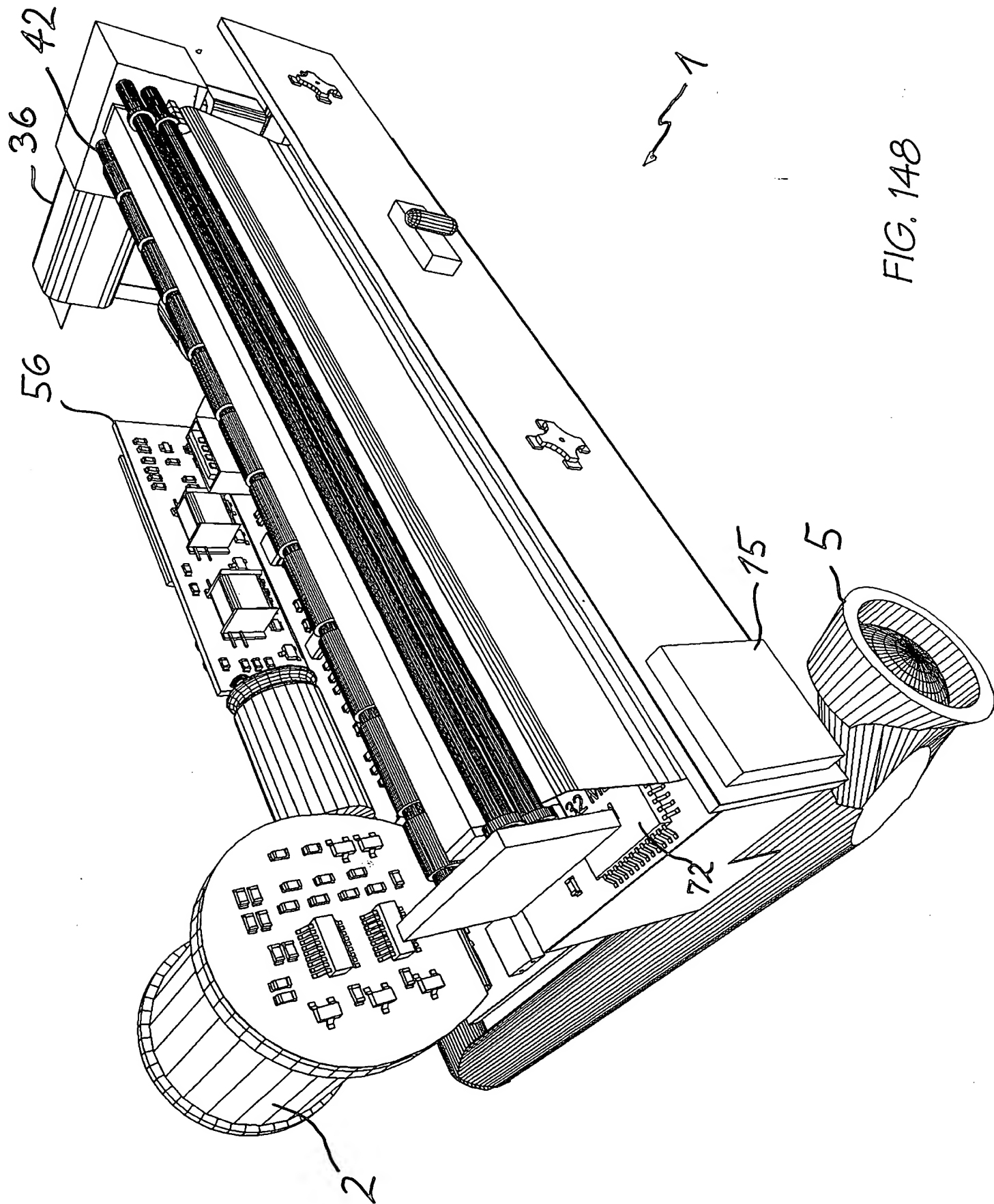


FIG. 148

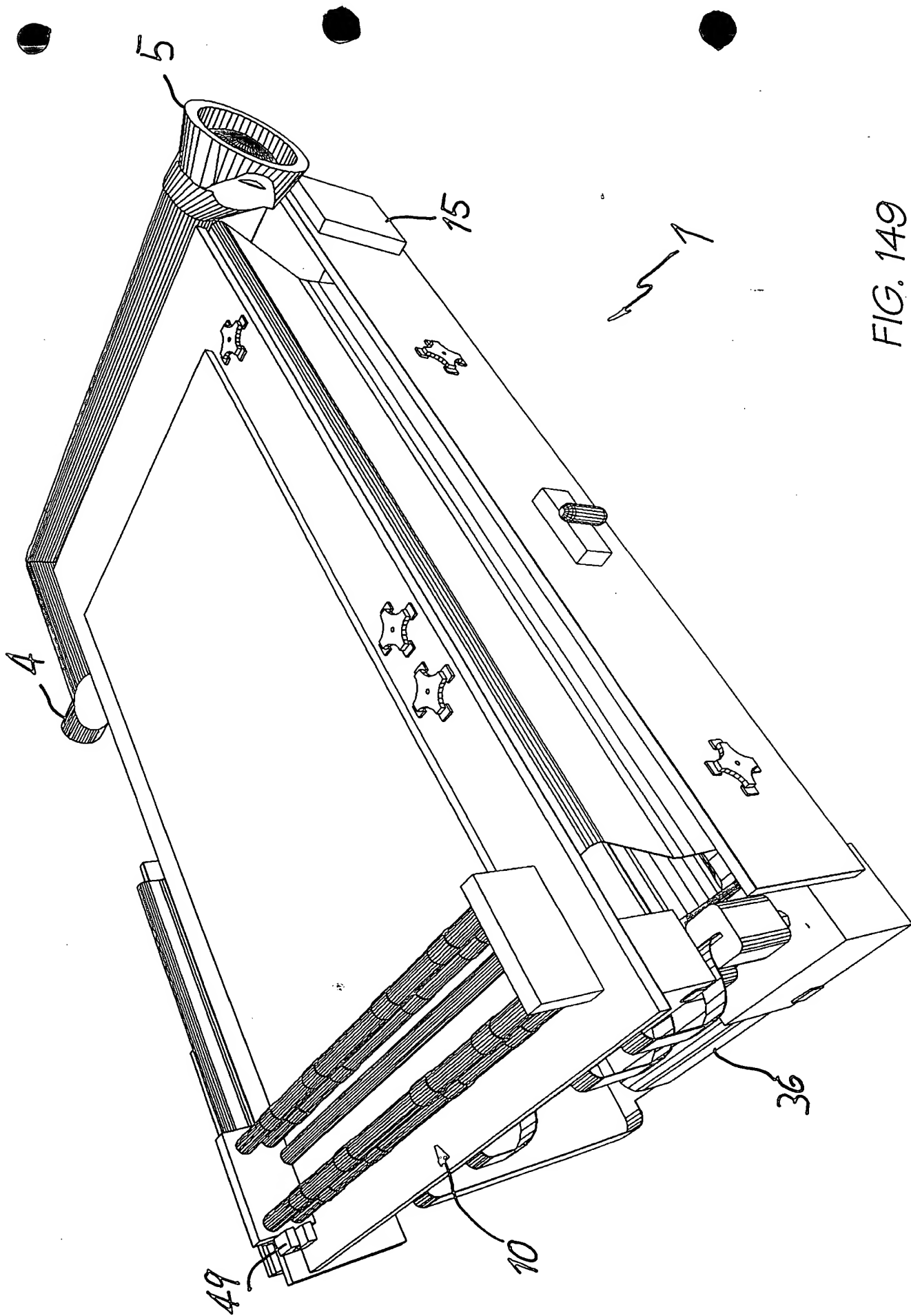


FIG. 149

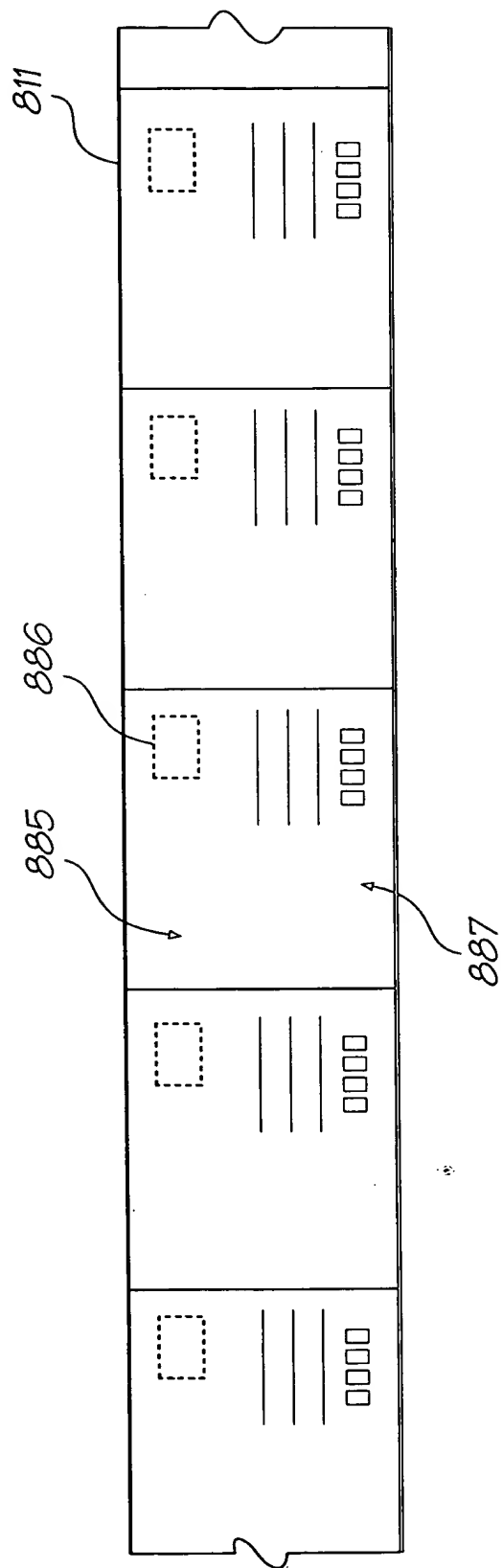


FIG. 150

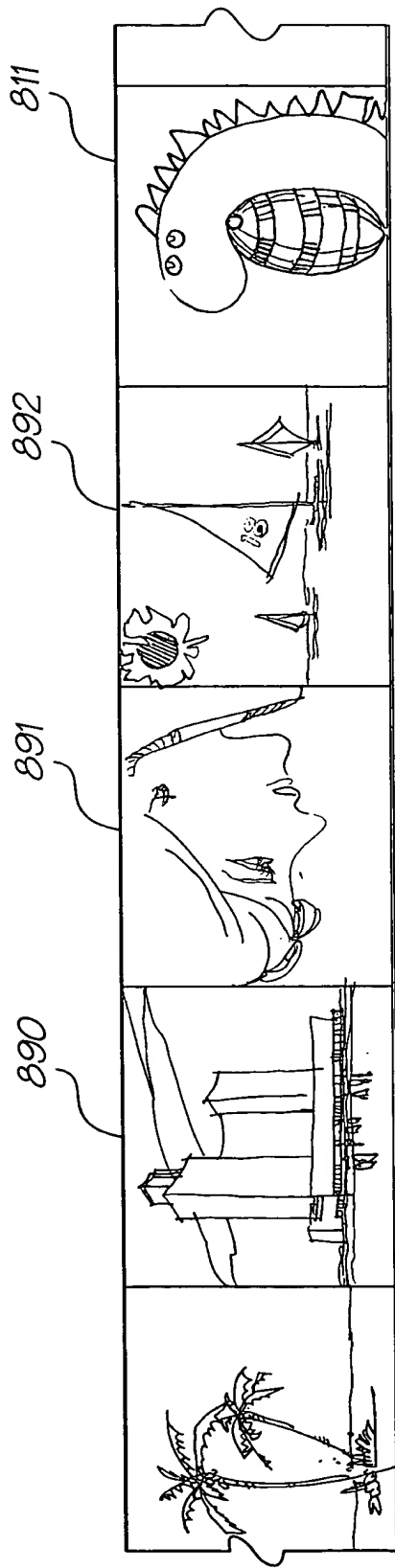


FIG. 151